UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

INGENIERÍA DE TELECOMUNICACIÓN

PROYECTO FIN DE CARRERA

Implementación de un Sistema de Adaptación y Reconocimiento de Locutor en un Dispositivo Portable.

CÉSAR DÍEZ SÁNCHEZ

MARZO DE 2010

PROYECTO FIN DE CARRERA

Implementación de un Sistema de Adaptación y Reconocimiento de Locutor en un Dispositivo Portable.

Autor:
César Díez Sánchez

Tutora:
Ascensión Gallardo Antolín
Dpto. Teoría de la Señal y Comunicaciones
Universidad Carlos III de Madrid

Marzo de 2010

不怕慢，就怕站

No importa la lentitud con que avances, siempre y cuando no te detengas.

Confucio

Agradecimientos

Aquí está, ¡por fin! Después de tanto esfuerzo aquí lo tenéis, terminado de una vez. En primer lugar y como no podía ser de otra manera, este trabajo terminado quería agradecérselo a toda mi familia, sobre todo a mis padres que me han dado la oportunidad de ser lo que en el futuro pueda llegar a ser, sin vuestra ayuda nada hubiera sido igual, aquí lo tenéis, va por vosotros.

No quiero dejar pasar la oportunidad de dar las gracias a todos aquellos que habéis contribuido en mayor o menor medida. A todos los que forman parte del Departamento de Teoría de la Señal y Comunicaciones y muy especialmente a Ascensión, una estupenda profesora y tutora que siempre ha estado dispuesta a ayudarme en todo lo que he necesitado y me ha facilitado muchísimo el trabajo que aquí finaliza. A toda la gente que trabaja dentro del Grupo de Procesado Multimedia, particularmente, a Ana Isabel y su paciencia para tratar de explicarme todas mis dudas. Gracias.

A todos los compañeros que he tenido durante la carrera desde hace ya unos cuantos años. A Myriam, porque me ayudó mucho conocerte al principio. A Diego, por tantas y tantas cosas que no me pueden caber aquí, pero tú lo sabes. A Inés, por aportarme mucha claridad en determinados momentos aunque seguramente no seas consciente. A Irene, por ser la mejor compañera de prácticas que he tenido. A Blanca, porque también han sido muchos momentos y muchas charlas de gran ayuda, y porque vives cerca de mí también, jeje. A Salva, Omar y Luis, lo siento chicos, pero es que vosotros sois un trío indivisible, por hacerme pasarlo tan bien en tantos momentos. Y también a Elvira, Cris, Telmo, Duarte… es imposible nombraros a todos, que nadie se enfade. Gracias a todos.

A tanta y tanta gente que conocí en ese año maravilloso en Finlandia, a todos mis Tampereños, muchas gracias. A Juanlu, Pablo, Dan, Claudia, Rodrigo, María, Jorge y tantos otros. Gracias

Por supuesto, y no por estar aquí cerca del final van a ser menos importantes, a mis amigos de siempre, porque vosotros de verdad sois los que mejor me conocéis y creo que poco más puedo decir, por estar siempre ahí, porque son muchos años ya aguantándome, por las veces que os he tenido que decir que no a cualquier cosa y porque esto lo celebramos! Aquí quiero mencionar a Elena, porque sabes un poco más lo que es esto de teleco, y eso ayuda, porque en los primeros años me ayudaste muchísimo más de lo que crees y porque así no era el único raro del grupo, jeje. Aquí lo tenéis, por fin está acabado y una parte es vuestra, gracias a todos.

Y para terminar, a Anne, por apoyarme independientemente de todo, por aguantarme, porque sé que todo esto te suena a chino pero intentas entenderlo, porque sé todos los esfuerzos que haces, por todo el tiempo que no hemos aprovechado, porque me haces ser mejor persona y porque eres genial. Esto no hubiera quedado igual sin ti, gracias.

Gracias a todos.

Resumen

En el presente documento se describe un proyecto que se encuentra situado dentro del marco de las Tecnologías del Habla. En el mismo, se desarrollan varios módulos que se integran en el sistema de Verificación de Locutor independiente de texto diseñado para dispositivos portátiles del tipo PDA.

En la primera parte de este proyecto se han centrado todos los esfuerzos en conocer la descripción de este tipo de aplicaciones. Para ello, en el presente documento se detalla cada uno de los bloques de que el Verificador está compuesto. La segunda parte del presente trabajo se centra en la todo el proceso de implementación hasta lograr la versión final del programa. En esta segunda parte se mostrarán tanto resultados como conclusiones obtenidas. En la última parte se presentan los manuales de usuario de cada uno de los módulos implementados en la realización de este proyecto.

Dado que se trata de un proyecto muy extenso, se ha decidido dividir la aplicación presentada en diferentes módulos, todos ellos desarrollados en eVC++. En concreto, han sido cuatro los módulos desarrollados: un grabador que funciona en la PDA para crear una buena base de datos; un módulo de parametrización de ficheros de voz para extraer de las grabaciones las características del locutor; un tercer módulo adaptador que logra crear el modelo de usuario adaptado a las características propias de cada locutor y por último, el módulo de Verificación de Locutor que realiza la parte más importante de este proyecto.

En el Verificador, un usuario tratará de identificarse eligiendo su identidad a través de una lista de usuarios. Posteriormente, el usuario deberá grabar una locución durante un cierto periodo de tiempo para que el programa pueda determinar si el usuario se trata del locutor que dice ser o no.

El objetivo final de este Verificador no termina en sí mismo, sino que se trataría de una aplicación que se podría utilizar para dar acceso a un sistema protegido. De esta manera, se evitaría el acceso a través de contraseñas que pueden ser descifradas por algún impostor. En el caso que nos ocupa en este trabajo, el acceso estaría en las características del propio usuario, por lo que nadie más que él mismo sería capaz de acceder al sistema.

Índice general

Índice de imágenes

Índice de tablas

Índice de gráficas

CAPÍTULO I

Introducción

1.1 Marco

El siguiente Proyecto Fin de Carrera ha sido realizado en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid por el alumno César Díez Sánchez, dirigido y supervisado por la doctora Ascensión Gallardo Antolín, profesora titular del Departamento de Teoría de la Señal y Comunicaciones.

Este Proyecto se encuadra dentro de las líneas de investigación del Grupo de Procesado Multimedia del Departamento de Teoría de la Señal y Comunicaciones relacionadas con las tecnologías del habla.

1.2 Motivación

1.2.1 Contexto

Desde la década de los noventa, los dispositivos móviles se han convertido en un elemento fundamental de nuestra sociedad. Cada día las posibilidades que ofrecen estos dispositivos aumentan y se abren nuevas vías de negocio que emplean estos dispositivos.

El estrés de la vida moderna convierte cada instante en algo precioso y es por ello que surgen necesidades como trabajar desde cualquier parte, realizar operaciones bancarias a través de la red, acceder al correo electrónico, etc. Un factor determinante en el desarrollo de aplicaciones que cubran todos estos cometidos es la necesidad de seguridad.

1.2.2 Ataques a la privacidad

En un mundo conectado a la red, los proveedores de servicios de la Sociedad de la Información se enfrentan cada día más al problema de proteger sus sistemas y recursos de posibles ataques procedentes del entorno global al que se encuentran conectados.

La realidad es que la gran mayoría de los sistemas informáticos, sea cual sea su orientación y su función desde el punto de vista de servicio, basan su protección de acceso en la utilización de contraseñas. Y esto es así a pesar de la fragilidad, desde el punto de vista de la seguridad, que una autenticación basada en contraseñas puede ofrecer.

Todos nosotros, como ciudadanos en la Sociedad de la Información estamos expuestos a numerosos ataques cada vez más sofisticados, con el objetivo de apropiarse de estas contraseñas o claves 'secretas' que nos identifican ante los sistemas y servicios de nuestro entorno. Esta tendencia a 'capturar' nuestra identidad es creciente y hay numerosas técnicas de ataque para conseguir el 'phising' de las claves personales y otros datos identificativos con el fin de suplantar nuestra identidad en los servicios de información que nos rodean: SCAM, Ingeniería Social, pharming (dns poisoning), eavesdropping (sniffers, keyloggers), tampering, web spoofing, homograph, IDN spoofing, fake mail, looping, entre otros.

1.2.3 Alternativas

La seguridad objetiva de los sistemas de información y comunicaciones puede mejorarse de forma sustancial cuando se utilizan técnicas sofisticadas de identificación personal antes de permitir el acceso a los recursos electrónicos e informáticos de los sistemas y servicios. Estas técnicas están dirigidas a utilizar factores adicionales al tradicional basada únicamente en guardar un secreto (la contraseña), introduciendo factores adicionales como son dispositivos que están en manos del usuario cuando accede al sistema, o características biométricas del propio usuario.

En este sentido, las tecnologías que introducen las claves de acceso de un solo uso (One-Time-Password) basadas en *mecanismos de autenticación factor-2*[1] proporcionan una alta garantía de seguridad para el acceso a los recursos y sistemas.

Ya hay soluciones comerciales en el mercado de autenticación de factor-2. Muy recientemente se han portado estas técnicas a dispositivos móviles (típicamente teléfonos Java o PDAs) y que, portados por su usuario funcionan como token generador de claves de un solo uso.

Dando un paso más allá, en este Proyecto se plantea investigar la posible utilización de un dispositivo móvil como plataforma sobre la que desarrollar un sistema de autentificación multifactor y multimodal: sumamos a una palabra de paso estática (algo que el usuario sabe), un dispositivo portable (algo que el usuario lleva) capaz de albergar una aplicación (token) que genere palabras de paso con una validez temporal muy reducida (1 minuto), y una identificación biométrica mediante voz (algo que el usuario es), también soportada por el dispositivo portable.

El habla es una herramienta básica de comunicación entre los seres humanos y podría considerarse como el medio natural mediante el que interaccionamos entre nosotros. El desarrollo de aplicaciones informáticas capaces de comprender y sintetizar el habla supone un paso hacia adelante en el proceso de facilitar la interacción hombre-máquina.

[1] *mecanismos de autenticación factor-2* son los que están basados en algo que sólo el usuario sabe, su PIN, junto con algo que sólo el usuario posee, un dispositivo generador de claves

Las señales acústicas generadas por cada locutor son diferentes y esta característica biométrica convierte a la voz en un elemento muy importante para la identificación de personas.

Una ventaja añadida que aporta este tipo de aplicaciones basadas en la voz es la libertad que proporcionan al usuario, por ser sólo necesario el uso de la voz para su manejo.

Esta identificación biométrica podría completarse, utilizando los nuevos dispositivos móviles diseñados para videoconferencia, con una identificación visual.

1.3 Objetivos

En este entorno en el que nos hemos situado es donde se ubica el objetivo de este proyecto, donde la combinación de las tecnologías del habla y los dispositivos móviles tienen una importante vía de aplicación.

De modo que el objetivo es implementar un sistema de *verificación de locutor independiente del texto*[2] para su posterior utilización en dispositivos portátiles como puedan ser, por ejemplo, teléfonos móviles o PDAs.

Para alcanzar la solución final consistente en el sistema verificador, se ha tenido que solucionar el problema de portar diferentes aplicaciones para que puedan funcionar en los dispositivos móviles en estudio. El proyecto queda, en definitiva, dividido en varios módulos de manera que se organiza el sistema global de forma más eficaz y organizada.

El primero de los módulos de que consta este proyecto, se trata de un sistema de grabación de voz para PDA en el cual el usuario podrá grabar las frases de entrenamiento y poder formar así una base de datos.

Una vez obtenida la base de datos correspondiente se procederá a utilizar el siguiente módulo de parametrización, el cual se encarga de obtener las características de cada fichero de voz grabado para su posterior análisis.

Posteriormente, el siguiente módulo será el que se encargue de realizar la adaptación de modelos de usuario de manera que contenga las características biométricas del usuario. Este modelo de usuario servirá en el proceso de verificación de locutor para comparar la muestra de voz del locutor y determinar si la locución realizada se corresponde con el modelo elegido o no.

El último módulo en que queda dividido este proyecto es el verificador en sí, el cual no tendría sentido sin los módulos anteriores ya que el verificador necesita disponer de los modelos de usuario correctamente calculados.

Todos estos módulos se podrán instalar en una PDA en cuatro diferentes aplicaciones que contarán todas ellas con una interfaz amigable y sencilla para el usuario. Hay que tener en cuenta que el dispositivo ha de tener una capacidad de almacenamiento considerable para poder grabar las muestras de entrenamiento. De la misma manera, se ha tenido muy en cuenta la capacidad de procesamiento limitada que tienen estos dispositivos en comparación

[2] *verificación de locutor independiente del texto* es el proceso de verificación en el que no se tiene en cuenta qué dice el locutor para verificarle, en contraposición con la dependiente de texto.

con un PC. Debido a esta limitación ha sido necesario optimizar bien los recursos y se ha tenido que pasar de aritmética en *punto flotante*[3] a una en *punto fijo*[4], lo cual no ha sido sencillo.

1.4 Estructura de la memoria

La memoria del proyecto fin de carrera "Implementación de un Sistema de Adaptación e Identificación de Locutor en un Dispositivo Portable" se presenta estructurada en capítulos. A continuación se detalla el contenido de todos ellos, de forma que sirva de guía para el lector.

- *Capítulo 2: Estado del arte*

  Este capítulo presenta una perspectiva histórica de las tecnologías del habla, así como un resumen de las principales técnicas empleadas en la actualidad para el procesado del habla, centrándose en las tecnologías empleadas en la verificación de locutor.

- *Capítulo 3: Descripción del sistema*

  En este capítulo entraremos a describir los diferentes módulos en que se ha dividido este proyecto. Se explicarán con detalle todos los procedimientos necesarios para conseguir que los sistemas de grabación, parametrización, adaptación y verificación funcionen correctamente.

- *Capítulo 4: Implementación en PDA*

  Quedará detallada cómo ha sido la implementación final de los programas para poder portarlos a la PDA. En este capítulo se explicará también el funcionamiento de cada uno de los módulos.

- *Capítulo 5: Pruebas experimentales*

  En este capítulo se mostrarán una serie de resultados finales en función de diferentes configuraciones de los módulos de modo que podamos obtener una comparativa entre ellas. Con la comparación de los resultados indicados se procederá a efectuar un análisis de la aplicación diseñada y su solución final.

- *Capítulo 6: Conclusiones y líneas futuras*

  Este capítulo refleja la síntesis de los resultados obtenidos tras la realización del proyecto. Se destacarán los objetivos alcanzados así como las problemáticas no resueltas. Por último se indicarán las líneas por las que podrían discurrir futuras investigaciones para la mejora de la aplicación implementada.

- *Capítulo 7: Presupuesto*

  Se reflejará una valoración del presupuesto necesario para el desarrollo del proyecto. Para el cálculo del mismo se tendrán en cuenta los costes del material así como los costes de los honorarios de los desarrolladores de la aplicación.

[3] *Punto Flotante* es un método de representación de números reales que se puede adaptar al orden de magnitud del valor a representar. Se utiliza cuando se calculan funciones que requieren precisión fraccionaria. Los cálculos complejos, como la raíz cuadrada, o trigonométricas, como el seno y el coseno, tienen como resultado un valor cuya precisión requiere un tipo en coma flotante.

[4] Los sistemas en *Punto Fijo* consisten en destinar una cantidad fija de dígitos para la parte entera y otra para la parte fraccionaria. La cantidad de dígitos destinados a la parte fraccionaria indica en definitiva la posición de la coma dentro del número.

- *Apéndice Manuales de usuario*

En este apéndice se incluirán los diferentes manuales de usuario para los cuatro programas de que consta este Proyecto de modo que un usuario novel sea capaz de manejarse correctamente.

- *Bibliografía*

Por último, se incluye en el final del documento una mención a las referencias bibliográficas utilizadas para la realización del mismo.

CAPÍTULO II

Estado del arte

2.1 El Habla

2.1.1 Mecanismo de producción del habla

Para que se produzca cualquier sonido es necesario disponer de tres elementos esenciales:

- Un elemento que vibre.
- Un medio elástico, en el que se propaguen las vibraciones.
- Una caja de resonancia que amplifique las vibraciones.

El ser humano dispone de su aparato fonador, que consta a su vez de los siguientes conjuntos de órganos que realizan cada una de las funciones indicadas:

- Los pulmones y músculos respiratorios que generan la corriente de aire.
- Las cuerdas vocales, localizadas en la laringe, que vibran.
- La boca, la nariz y la garganta que funcionan como resonador.
- Los labios, dientes, paladar duro, velo del paladar y mandíbula que funcionan como articuladores.

En la siguiente figura se observa cada uno de los elementos del aparato fonador humano:

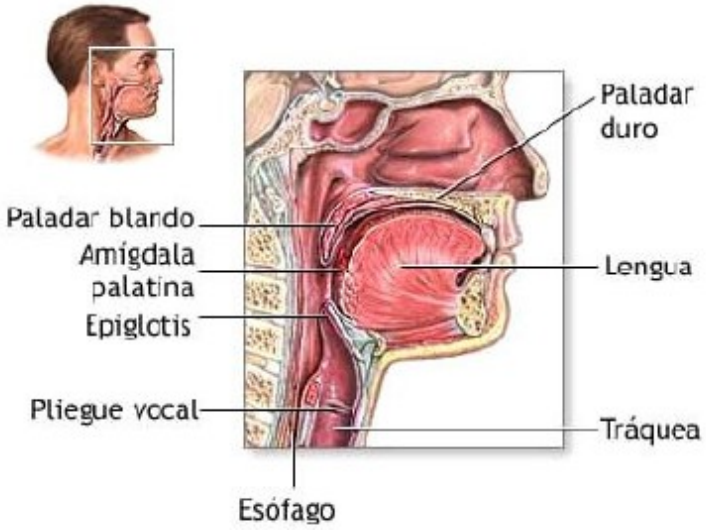


Imagen 2-1. Elementos del aparato fonador humano

El proceso del habla se inicia cuando se expira el aire contenido en los pulmones, la columna de aire atraviesa la tráquea y hace vibrar las cuerdas vocales. Es en la laringe donde se genera la voz en su tono fundamental y sus armónicos.

La voz sufre una modificación en la caja de resonancia en la que se amplifica y adquiere el timbre de voz. Finalmente los órganos articuladores moldean la columna de aire formando los fonemas, sílabas y palabras.

En la siguiente figura podemos observar el esquema del proceso de generación de la voz humana:

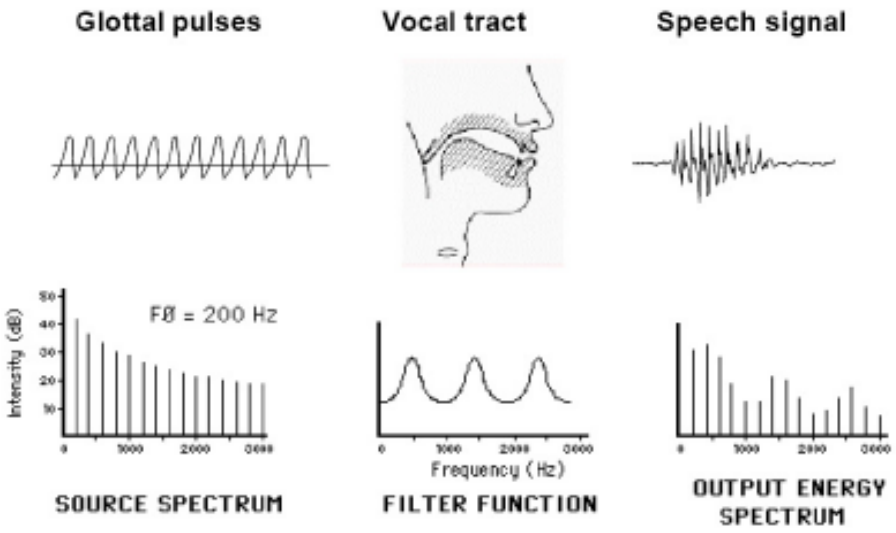


Imagen 2-2. Proceso de generación de la voz humana

2.1.2 Variabilidad del locutor

Las características del sistema fonador son propias a cada individuo. Lo que hace que la voz pueda ser empleada como un elemento biométrico, es lo que se denomina variabilidad interlocutor.

Pero así como la voz generada por un individuo difiere de la voz de los demás, no siempre la voz de un mismo individuo suena igual. Una misma persona no pronuncia siempre de la misma manera, debido a diferentes factores tanto físicos como psicológicos, es lo que se denomina variabilidad intralocutor.

La variabilidad intralocutor introduce una complejidad adicional a la problemática de la verificación de locutores. Si distinguir un locutor de otro es relativamente fácil, el relacionar una muestra de voz con el locutor que la generó es sustancialmente más complejo. Esta característica propia de la voz humana, supone una importante diferencia con otros sistemas biométricos, como la huella dactilar. Los principales factores que afectan a la variabilidad intralocutor son la edad del hablante, el estado de salud (enfermedades respiratorias), el estado de ánimo, etc.

Además de los factores citados, se deben destacar otros factores, externos al locutor, que dificultan el proceso de verificación de locutor. Éstos son el ruido acústico del entorno y el micrófono empleado.

2.2 Tecnologías del habla

2.2.1 Historia

Las tecnologías del habla se encuentran en pleno desarrollo e introducción en nuestro entorno cotidiano. El habla, medio de comunicación por excelencia entre seres humanos, comienza a ser utilizada como medio de comunicación con la tecnología desarrollada por el ser humano. La capacidad de interactuar con las máquinas mediante el habla ha sido una constante en el desarrollo tecnológico de la humanidad.

Durante siglos la humanidad se ha sentido admirada de una de sus más importantes capacidades, el habla. En el afán de lo lograr reproducir o imitar esta cualidad humana por medios mecánicos trabajaron muchos. En el siglo XVII, entre los precursores del estudio del habla humana se encontraba Athanasius Kircher aunque no pasó de imaginar teorías extrañas y los más raros instrumentos posibles [KIR50].

El siguiente paso consistía en realizar una máquina parlante. El primero en intentar tal proeza, hasta donde he podido conocer, fue el profesor de fisiología Ch. G. Kratzenstein, de Copenhague, allá por la segunda mitad del siglo XVIII, a través de un sistema de tubos de órgano, con los que se propuso reproducir las vocales.

Más o menos al mismo tiempo, en Viena, Wolfgang Von Kempelen [KEM91], en 1791, construyó un rudimentario sintetizador de voz capaz de pronunciar frases cortas mediante un ingenio mecánico.

Imagen 2-3. Sintetizador de Wolfgang Von Kempelen

El habla era algo que seguía esquivando a los constructores de aquellos primitivos antepasados de los robots. Aun así, se llevaron a cabo muchos intentos de construir máquinas de apariencia humana capaces de "hablar". En el siglo XIX se extendió el uso de máquinas similares a la de Kempelen, que no aportaron nada nuevo, hasta que llegó el británico Joseph Faber y se decidió a recrear fielmente el sonido del habla. Corría el año 1835 y su máquina causó asombro general. Se trataba de un artilugio de considerable tamaño, manejado a través de teclados y pedales, al que denominó *Euphonia*.

En este mismo siglo XIX se comenzaron a desarrollar las bases matemáticas en las que se basa el análisis de la señal de voz, como el análisis de Fourier. Durante este siglo se realizaron una serie de inventos que potenciaron la utilización del habla como método de comunicación. El teléfono permitió utilizar la voz para comunicaciones instantáneas y a larga distancia; el fonógrafo permitió grabar registros de voz, siendo lo que podríamos denominar el primer sistema de respuesta oral. Durante muchos años se intentaron simular las vías respiratorias humanas para mejorar este tipo de aparatos, como en el caso del artefacto perfeccionado por el estadounidense R.R. Riesz en 1937.

Pero la simple mecánica no podía dar más de sí. Entonces llegó la electricidad y lo cambió todo, pasando los pioneros de las máquinas parlantes a ingeniar sistemas de síntesis de voz por medios eléctricos. Las máquinas de fuelles perdieron la batalla de la voz. En el siglo XX, se presentó una versión electrónica del sintetizador de Kempelen. El sistema denominado VODER [DUD39] fue desarrollado por Homer Dudley en Estados Unidos en 1939 y consistía en un teclado muy elaborado que permitía controlar la articulación y generación de sonidos vocálicos y consonánticos.

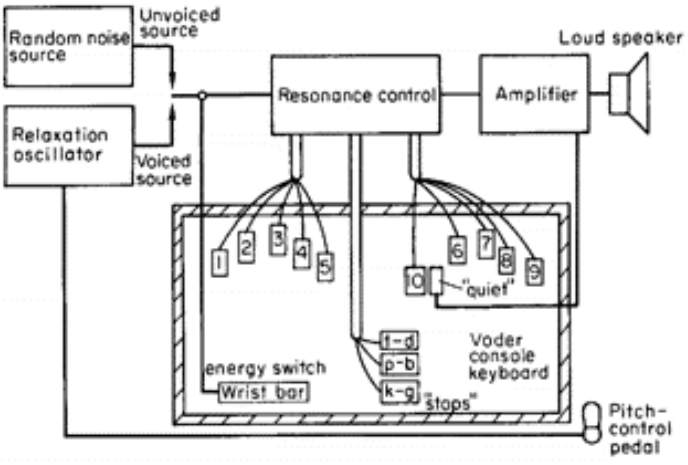


Imagen 2-4. Sintetizador de voz VODER

Este sistema pionero sembró las bases de los modernos sintetizadores de voz, basados actualmente en tecnología digital, pero compartiendo la misma teoría sobre el modelo de producción del habla. Ya desde los años 30 podemos decir que comenzó la investigación sobre la señal de voz con el desarrollo de sistemas prácticos de transmisión digital mediante modulación PCM.

Un hito muy importante dentro de la investigación y desarrollo de las tecnologías del habla fue el invento del espectrógrafo de voz en la década de 1940. El espectrógrafo de voz permitía obtener un registro gráfico de la evolución de la energía de la señal de voz en diferentes bandas frecuenciales. Esta visión tiempo-frecuencia de la señal de voz conjuntamente con los estudios sobre el sistema auditivo humano abrió el campo a la investigación en una de las áreas más fascinantes de las tecnologías del habla: el reconocimiento automático del habla. De hecho, en 1952 se presentó un sistema electrónico de reconocimiento de voz en los laboratorios Bell que era capaz de reconocer, dentro de un margen aceptable de error, los dígitos pronunciados en inglés.

La gran revolución en las tecnologías del habla se comenzó a gestar con el advenimiento de las computadoras digitales y la informática. Desde mediados de los años 60 se comenzaron a desarrollar algoritmos de procesado de señal, a desarrollar las bases de la inteligencia artificial y comenzaron a aparecer los primeros desarrollos e investigaciones en sistemas de reconocimiento automático del habla utilizando computadoras digitales.

Pero es en la década de los 70 cuando se da el impulso definitivo con la financiación de grandes proyectos de reconocimiento del habla por parte de países como EE.UU., Japón, Francia, Alemania, etc. En Noviembre de 1971 la Oficina de tecnología para el procesado de la información de la ARPA (Advanced Research Projects Agency of the USA Department of Defense) iniciaba un ambicioso proyecto de investigación con el objetivo de desarrollar varios sistemas de comprensión automática del habla continua para varios locutores cooperantes que hablasen "General American dialect" [LEA80].

Cinco años más tarde y una vez finalizado el proyecto inicial, el resultado final fue en cierta forma decepcionante ya que los objetivos no llegaron a alcanzarse pero sí que sirvieron de toma de contacto con la realidad del problema y sentaron las bases para posteriores investigaciones. Una de las conclusiones finales, en las que se trataba de evaluar la capacidad computacional necesaria para abordar con garantías el funcionamiento en tiempo real de estos sistemas, situaba la capacidad mínima necesaria alrededor de los 100 *MIPS*[5] [KLA77].

El requisito de disponer de una capacidad computacional de este orden con sistemas informáticos de bajo coste (Ordenadores Personales, Estaciones de Trabajo...) no se ha alcanzado hasta la actualidad, y aún de una forma parcial, gracias a la aparición en el mercado de forma masiva de la tercera generación de microprocesadores especializados en el procesado digital de señal (DSP-Digital Signal Processors)[LEE88]. Los avances en la tecnología de integración de circuitos integrados (VLSI), la aportación de nuevos conceptos en arquitectura de computadores (Arquitecturas RISC y Harvard) junto a las decisiones de diseño para operar en un entorno de tiempo real incentivaron la aparición de DSP de tercera generación.

Esta revolución tecnológica de la década de los 80 y principios de los 90, conjuntamente con la investigación básica realizada durante esta década por multitud de grupos de investigación en todo el mundo en el campo de las tecnologías del habla y de la inteligencia artificial han construido los cimientos del desarrollo actual en sistemas de comunicación oral hombre-

[5] *MIPS:* Million Instructions per Second (Millón de Instrucciones por Segundo).

máquina, encontrándonos en las puertas de una gran revolución en el desarrollo de sistemas basados en tecnologías del habla.

2.2.2 Ámbitos de aplicación

Las tecnologías del habla tienen como objetivo aplicar los conocimientos que se poseen sobre el lenguaje hablado, al desarrollo de sistemas informáticos. Los campos de aplicación de estas tecnologías son muy diversos pero se puede hacer una clasificación general en dos grandes grupos:

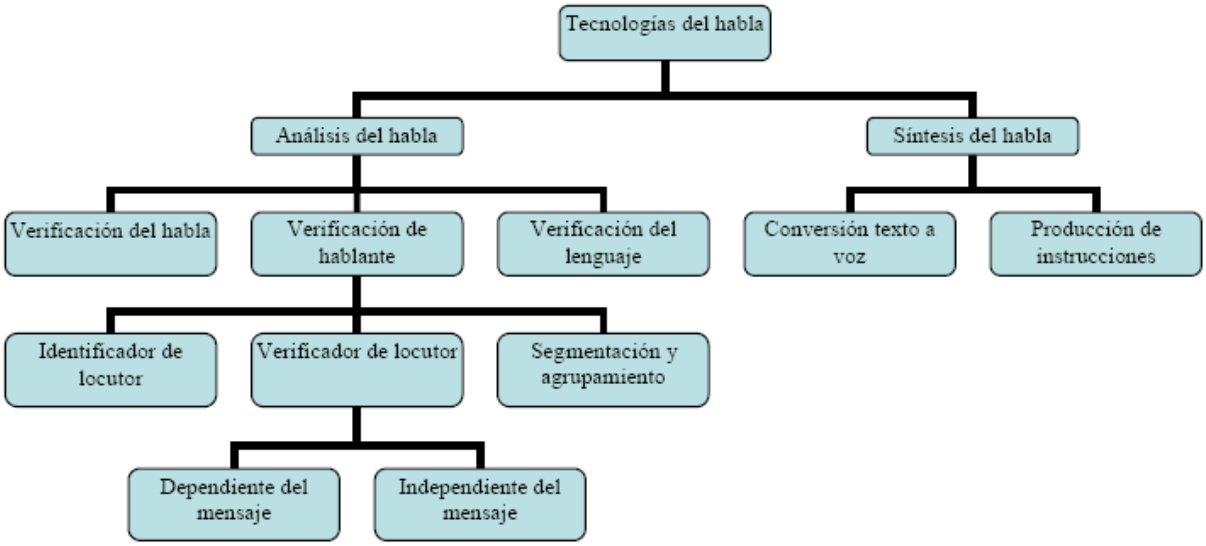


Imagen 2-5. Ámbitos de aplicación de las tecnologías del habla

Las tecnologías de **análisis** persiguen extraer información útil del habla, posibilitando a los seres humanos dar instrucciones a máquinas entre otras funciones. Por otro lado, la **síntesis** de habla abre una nueva vía de comunicación máquina-hombre más cómoda e intuitiva para éste último. En este proyecto nos centramos en la parte de análisis del habla que es donde se sitúa el verificador implementado.

2.2.2.1 Análisis del habla

Los seres humanos generan sonidos en forma de ondas de presión. Para poder ser procesadas deben ser codificadas en formato digital. El desarrollo actual de las tecnologías del habla permite que una máquina analice una señal de voz y extraiga información de ella, procese dicha información, genere una respuesta y finalmente, que la sintetice en forma de onda de presión.

Este proyecto se encuentra dentro de las tecnologías aplicadas al análisis del habla. Dada una señal de voz, tres tipos de información se pueden extraer de ella:

- Verificación del habla: ¿qué se dice?
- Verificación del lenguaje: ¿en qué idioma se dice?
- Verificación de hablante: ¿quién habla?

Como se ve en la Imagen 2-5, dentro de los sistemas de **verificación de hablante**, podemos encontrarnos con tres escenarios de aplicación: seguimiento y agrupamiento de locutores; identificación de locutor y verificación de locutor.

2.2.2.1.1 *Segmentación y agrupamiento de locutores*

Consiste en etiquetar qué locutor está hablando en un segmento de voz y cuándo se producen cambios de locutores. La segmentación y agrupamiento de locutores tiene su utilidad en la transcripción de noticias o reuniones, con el fin de aislar la voz de cada uno de los locutores en una grabación.

2.2.2.1.2 *Identificación de locutor*

El locutor no aporta información sobre su identidad y es el sistema el que determina quién es a partir de su voz dentro de un conjunto de posibles candidatos o, si se trata de identificación en conjunto abierto, si el locutor es conocido o no por el sistema.

Para cada usuario se dispone de un modelo de hablante y la aplicación lo que hace es calcular qué modelo es más probable que haya generado la muestra de voz a analizar. La identificación de locutores se puede utilizar para restringir el acceso a información a personas no autorizadas.

La siguiente figura muestra el esquema de funcionamiento de un identificador de locutor:

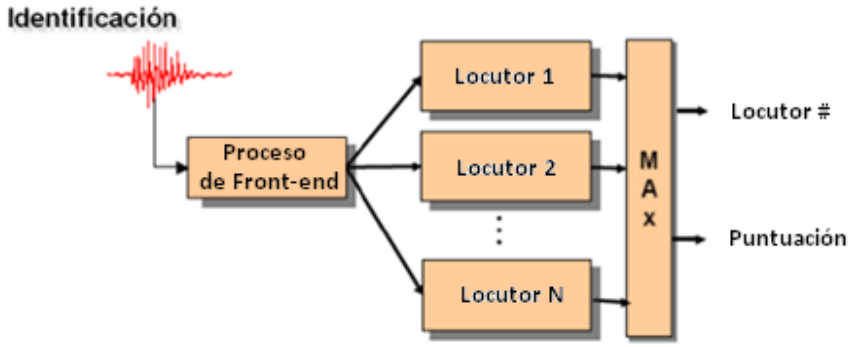


Imagen 2-6. Funcionamiento de un identificador de locutor

2.2.2.1.3 *Verificación de locutor*

Los sistemas de verificación difieren de los de identificación en que su tarea es determinar si el locutor es o no quien dice ser. Para cada usuario se dispone de un modelo de hablante y además se dispone de un modelo de mundo o impostor. La aplicación debe decidir si la muestra de voz es más verosímil que fuera generada por el usuario indicado o por el modelo de mundo.

La verificación de locutor presenta una menor complejidad que la identificación de hablante, lo que produce que la primera técnica tenga un mayor número de aplicaciones y una menor tasa de error.

La siguiente figura muestra el esquema de funcionamiento de un verificador de locutor:

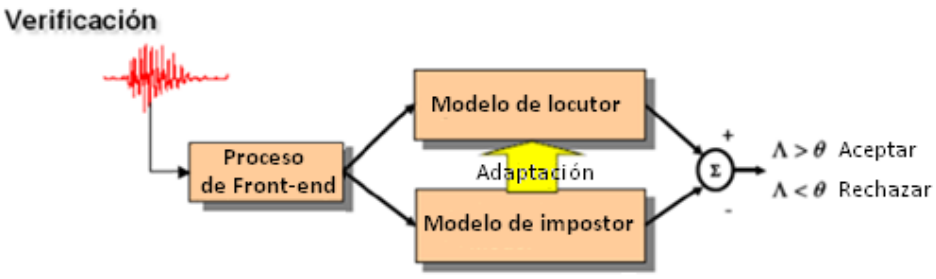


Imagen 2-7. Funcionamiento de un verificador de locutor

La verificación de locutor tiene numerosas aplicaciones comerciales importantes, por ejemplo, las transacciones bancarias a través del teléfono. Además de esta, existen muchas otras aplicaciones comerciales, todas destinadas a aumentar la seguridad en la verificación de la identidad. Una de ellas podría ser la gestión de identidad en centrales de atención al cliente, haciendo posible confirmar la identidad del usuario que llama y certificar las operaciones que realice, como la contratación o baja de nuevos servicios. Muy importante también son las aplicaciones en el ámbito forense, puesto que se puede emplear en juicios para comprobar si la voz empleada como prueba coincide con la del acusado.

Nosotros nos vamos a centrar en la **Verificación de Locutor** que se divide en dos categorías según la modalidad de discurso siguiendo la clasificación de las tecnologías del habla anteriormente mostrada. Éstas son:

- *Dependiente del mensaje*

  En este tipo de sistemas, la locución de entrenamiento y la de verificación suelen ser el mismo texto. Fundamentalmente consiste en una palabra o frase clave (contraseña) que le permite el acceso al sistema al usuario. En estos sistemas la contraseña es conocida por el sistema y suele ser fija, requiriendo un nuevo entrenamiento cada vez que se desea cambiar de contraseña.

  Un problema de estos sistemas es que son relativamente fáciles de atacar en caso de que el impostor grabe la palabra clave pronunciada por el usuario. Para evitar este tipo de ataques se introducen los sistemas "text-prompted" o de texto solicitado, en los que el sistema además de solicitar la contraseña al usuario solicita repetir un código o frase elegido aleatoriamente, y que por tanto evita la posibilidad de utilizar grabaciones.

- *Independiente del mensaje*

  En los sistemas independientes de texto, la locución de entrenamiento y la de test no coinciden, siendo la locución de test desconocida por el sistema. En este caso, el sistema no utiliza ningún tipo de contraseña, sino únicamente el rasgo biométrico de la voz.

  Estos sistemas presentan las desventajas de necesitar mayor cantidad de datos de entrenamiento y de proporcionar unas tasas de reconocimiento menor, debido en gran medida al aumento de la probabilidad de falso rechazo.

Ambas tareas son distintas y emplean por ello diferentes técnicas. En sistemas independientes de texto se utilizan tradicionalmente técnicas basadas en GMM (Gaussian Mixture Models), mientras que en sistemas dependientes de texto se suelen utilizar técnicas de DTW (Dynamic Time Warping) o HMM (Hidden Markov Models). El proyecto que en esta memoria se describe se encuentra en la segunda categoría, en la que corresponde a un verificador de locutor independiente del mensaje.

2.3 Etapas de un sistema de verificación de locutor

Todo sistema de Verificación de Locutor consta de varias fases que podríamos resumir a modo de esquema en la siguiente figura:

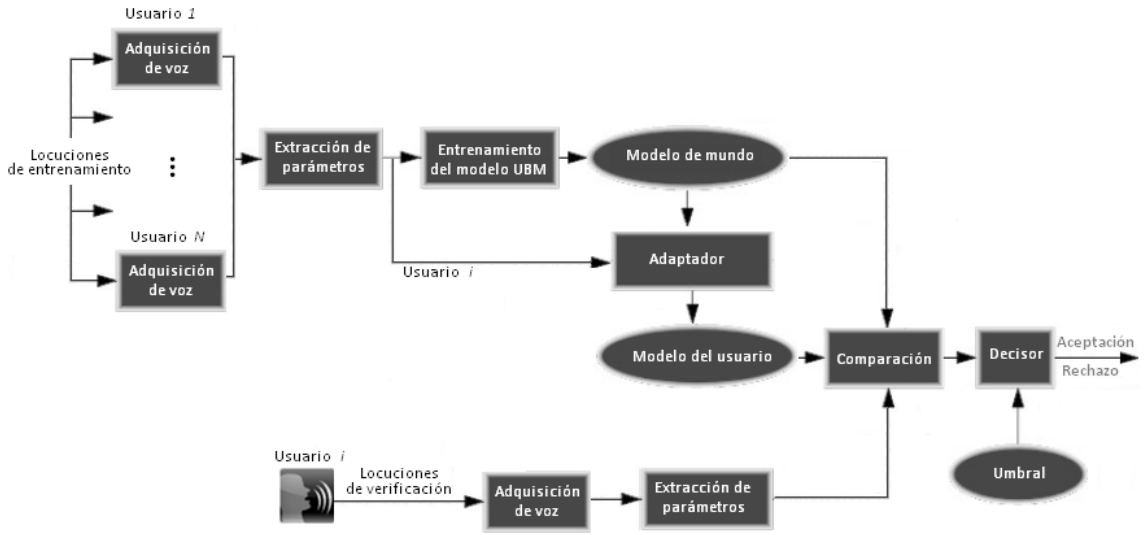


Imagen 2-8. Etapas de un sistema de verificación de locutor

Vista la figura podemos decir que las etapas de un sistema de verificación de locutor son:

- Adquisición de voz
- Extracción de parámetros
- Fase de entrenamiento (obtención de los modelos de mundo y de usuario).
- Fase de verificación (incluiría el cálculo de puntuaciones junto con la toma de decisión).

2.3.1 Adquisición de voz

La adquisición de voz se lleva a cabo mediante un micrófono o un auricular telefónico, que convierten la onda acústica en una señal analógica. A esta señal analógica se les aplica un filtro anti-aliasing para limitar el ancho de banda de la señal a la frecuencia de Nyquist. La señal analógica se muestrea para convertirla en una señal digital con un convertidor analógico/digital. Hoy en día los convertidores A/D para aplicaciones de voz muestrean habitualmente a una tasas de 8000 hasta 44100 muestras por segundo con una resolución de 12 a 16 bits por muestra.

En aplicaciones locales de verificación de locutor, el canal analógico es simplemente el cable y el acondicionamiento de la señal analógica. Es por esto que la señal digital resultante puede llegar a ser de alta calidad, sin las distorsiones que se producen al transmitirse las señales por las líneas telefónicas.

2.3.2 Extracción de parámetros

Una vez obtenida la señal digital se procede a la extracción de parámetros. Para ello se realiza un análisis espectral. Normalmente se obtendrán los Mel Frequency Ceptstral Coefficients (Coeficientes Cepstrales en Escala de Mel) que son coeficientes para la representación del habla basados en la percepción auditiva humana.

Se derivan de la Transformada de Fourier (FT) y de la Transformada Discreta del Coseno (DCT). La diferencia básica entre FT y MFCC es que en MFCC las bandas de frecuencia están espaciadas logarítmicamente (según la escala Mel) para modelar la respuesta auditiva humana más apropiadamente que las bandas espaciadas linealmente de la FT. Esto permite un procesado de datos más eficiente, por ejemplo, en compresión de audio.

La imagen siguiente representa el procesado de la señal que se realiza en un sistema típico para computar los coeficientes MFCC:

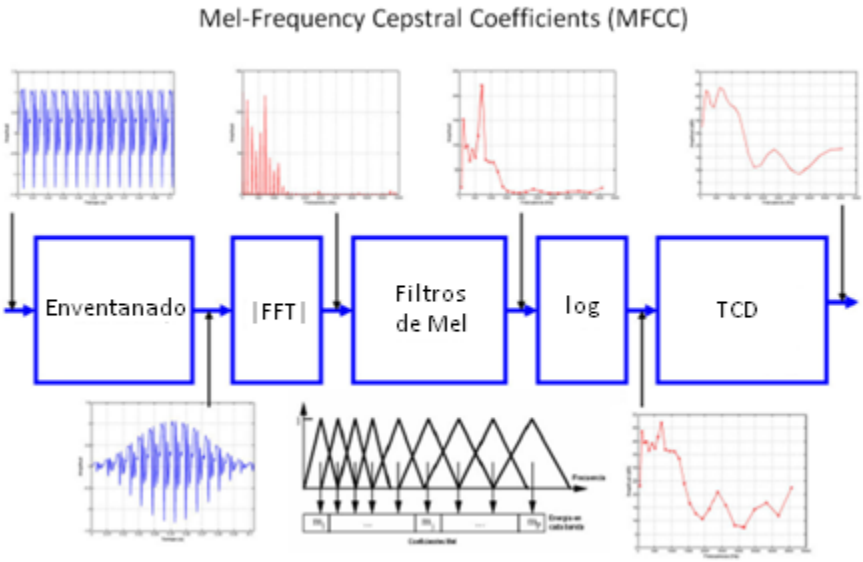


Imagen 2-9. Diagrama de bloques del proceso de cálculo de los MFCC

La señal acústica, muestreada a 8Khz en el caso de señales telefónicas, se diferencia (filtro de preénfasis) y se divide en un numero de segmentos solapados (windowing –enventanado-), por ejemplo, cada segmento de 25 ms de longitud solapados 15 ms entre sí.

A continuación se pasa al dominio de la frecuencia calculando la FFT (Fast Fourier Transform o Transformada Rápida de Fourier en español). Una vez en el dominio de la frecuencia la señal se filtra mediante un banco de filtros de diferentes frecuencias y amplitudes para dar más resolución en las bajas frecuencias, como ocurre en el sistema auditivo humano.

De la salida de cada filtro se calcula la energía en promedio (para la ventana) y los valores obtenidos se pueden ver como una nueva señal en tiempo discreto. Así, por ejemplo, usando un banco de 40 filtros se obtiene, para cada trama de voz de 25 ms un vector de 40 coeficientes. Se le aplica una transformación no lineal (aplicando el logaritmo neperiano) y por último se aplica una DCT. En este punto se obtienen unos parámetros (de los que se toman habitualmente de 13 a 20) aproximadamente incorrelados entre ellos: éstos son los coeficientes MFCC.

En particular, el primer coeficiente (C_0) representa la energía de la señal y se usa o no dependiendo de la aplicación (en caso de usarlo habitualmente se normaliza para compensar variaciones de energía debidas a proximidad al micrófono u otros efectos colaterales indeseados). Además de estos primeros coeficientes se suelen usar también las velocidades y/o aceleraciones, que representan la evolución temporal de los fonemas al pasar de unos a otros (Delta-MFCC y Delta-Delta-MFCC). Los coeficientes Delta representan la variación de los coeficientes MFCC alrededor del instante de tiempo considerado. Suelen, por esto, llamarse coeficientes de primera derivada o velocidad. De modo similar, los Delta-Delta se denominan de aceleración.

2.3.3 Fase de entrenamiento

Una vez que se han realizado las fases de adquisición de voz y de extracción de parámetros se procede al entrenamiento del modelo de referencia. Dicho entrenamiento se realiza en los HMM y GMM mediante distintas técnicas como la adaptación de modelos independientes de locutor. En el caso de DTW, el entrenamiento no es más que el almacenamiento de las locuciones (previamente parametrizadas) para usarlas como plantillas en la fase de verificación.

En el tema siguiente explicaremos más detalladamente las técnicas que se emplean en HMM y GMM para la obtención de los modelos de referencia.

2.3.4 Fase de verificación

Para terminar se realiza la verificación. Para ello se realizara un cálculo de puntuaciones entre el modelo de referencia obtenido con las elocuciones de test y los parámetros obtenidos de la elocución de verificación.

Una vez se tiene la puntuación se toma la decisión de aceptar o rechazar al usuario en el sistema en base a un umbral que podrá ser fijo o calculado para cada locutor.

Hay que destacar que en este punto hemos decidido saltar de modo intencionado la descripción detallada del entrenamiento y del cálculo de puntuación porque estos dos aspectos, que se abordaran en el tema 5 son muy dependientes de la técnica empleada (DTW, HMM, GMM) mientras que la toma de decisiones y la evaluación es independiente de la técnica empleada.

2.3.4.1 Marco genérico de la toma de decisión

Dado un segmento de voz X y un locutor S, el objetivo de la verificación de locutor es determinar si S generó la locución X. Esto se puede formalizar como un test de hipótesis básico entre las siguientes hipótesis:

H_0: X fue pronunciado por el locutor S.

H_1: X no fue pronunciado por el locutor S.

La decisión, de acuerdo con el criterio de máxima verosimilitud (Maximum Likelihood, ML) se obtiene mediante el cociente de verosimilitudes que viene dado por:

$$\frac{P(X \mid H_0)}{P(X \mid H_1)} \begin{cases} \geq \theta & \text{aceptar } H_0 \\ < \theta & \text{rechazar } H_0 \end{cases} \tag{2.1}$$

Donde $P(X|H_i)$, i=0,1 es la probabilidad de la hipótesis H_i evaluada por el segmento de voz X. θ es el umbral de decisión para aceptar o rechazar H_0. En principio debería ser 0, pero en aplicaciones prácticas interesa ajustar dicho umbral para controlar la relación entre las probabilidades de cometer errores en los dos sentidos posibles en la decisión. Habitualmente se suele utilizar el logaritmo del cociente anterior:

$$\Lambda(x) = \log P(X \mid H_0) - \log P(X \mid H_1) \quad \begin{cases} \geq \log\theta & \text{aceptar } H_0 \\ < \log\theta & \text{rechazar } H_0 \end{cases} \tag{2.2}$$

Por tanto, el objetivo de los sistemas de reconocimiento de locutor es encontrar métodos para calcular ambas probabilidades $P(X|H_0)$ y $P(X|H_1)$.

La forma de estimar estas probabilidades depende mucho de la aplicación y es un paso crucial en la implementación del detector. Para reconocimiento de locutor independiente de texto no existe información a priori de lo que el locutor ha dicho y en estas circunstancias la elección más acertada es el uso de GMM. Por el contrario, para reconocimiento de locutor dependiente de texto sí que existe tal información y por tanto, se suelen utilizar HMM de manera que se puede incluir esta información temporal adicional.

2.3.4.2 Errores en la decisión

En la verificación de locutores se pueden dar dos tipos distintos de errores:

- Falso Rechazo (FR): se produce cuando un usuario autentico es rechazado por el sistema
- Falsa Aceptación (FA), que aparece cuando un impostor es aceptado por el sistema como si fuera un usuario autentico.

Si se observa la distribución de las puntuaciones de usuarios e impostores se puede observar que, de manera general, ambas distribuciones se solapan, lo que supone un problema para seleccionar el umbral a partir del cual las puntuaciones serán interpretadas como pertenecientes a usuarios registrados. A continuación vemos este efecto en la figura:

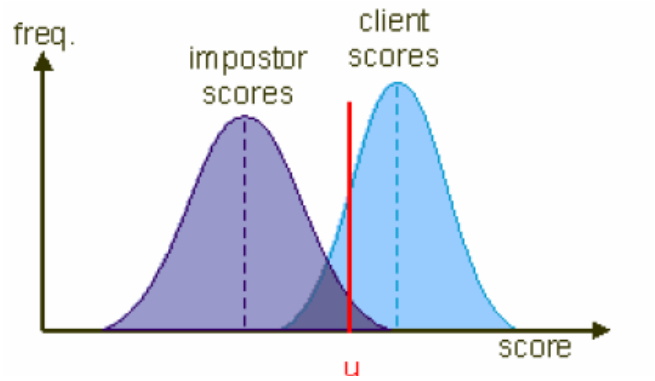


Imagen 2-10. Distribuciones de usuarios registrados e impostores

Por lo tanto, el área bajo la curva de impostores que queda por encima del umbral es la probabilidad de que un impostor sea aceptado. Esta probabilidad es la de Tasa de Falsa Aceptación (o False Acceptance Rate, FAR en sus siglas en inglés). La probabilidad de que un usuario registrado no sea aceptado también existe y es la equivalente al área que queda debajo del umbral bajo la curva de clientes registrados. Esta área es equivalente a la Tasa de Falso Rechazo (o False Rejection Rate, FRR en sus siglas en inglés).

En la siguiente figura vemos como varían la FAR y la FRR según situemos el umbral de toma de decisión:

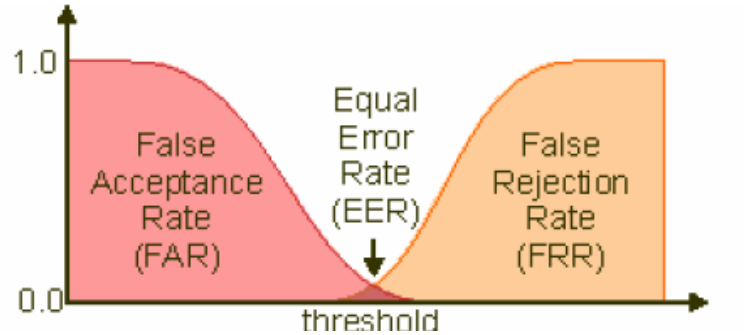


Imagen 2-11. Curva de la Tasa de Falsa Aceptación frente a la Tasa de Falso Rechazo

Si las distribuciones de puntuaciones de usuarios e impostores se solapan, la FAR y la FRR tendrán un punto de intersección, en el cual la FAR y la FRR son iguales. A este punto se le denomina Tasa de Error Equivalente (Equal Error Rate, ERR en sus siglas en inglés). Este punto se utiliza para comparar distintos sistemas y es donde el error del sistema, dado como la suma de la FAR y la FRR se suele minimizar. Sin embargo, para poder comparar dos sistemas según el ERR es necesario que éste sea calculado sobre los mismos datos de test utilizando el mismo protocolo experimental.

Como el ERR no describe plenamente el rendimiento del sistema, éste se suele representar mediante las curvas ROC (Receiver Operating Curve) y las curvas DET (Detection Error Tradeoff). En ambas curvas se muestra la Tasa de Falsa Aceptación frente a la Tasa de Falso Rechazo para distintos niveles de umbral. Las curvas DET se obtienen a partir de las curvas ROC realizando una transformación no lineal en los ejes, de manera que las curvas no lineales de las ROC se convierten en casi rectas. Esto las hace más sencillas de analizar y comparar unas con otras, por lo que se suele optar por las curvas DET.

En la siguiente figura podemos ver un ejemplo de ambas curvas:

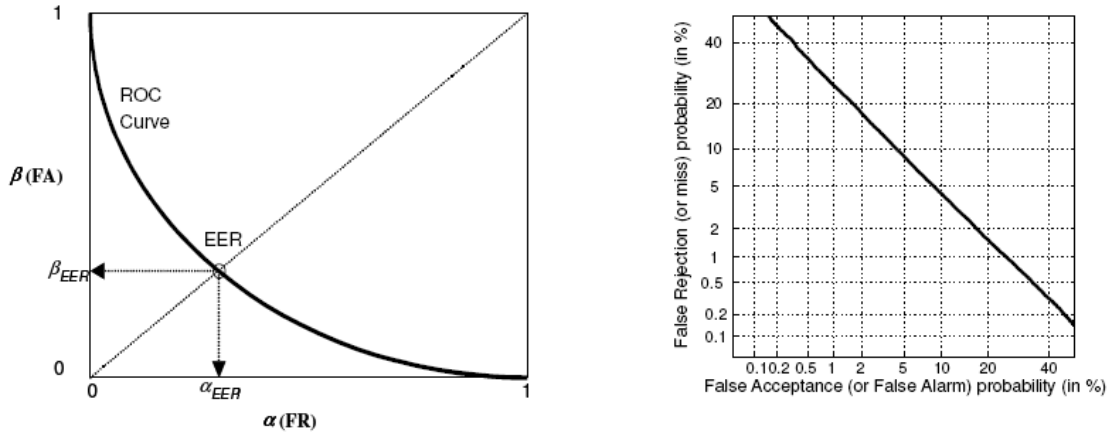


Imagen 2-12. Ejemplo de curvas ROC y DET

2.4 Técnicas empleadas para la verificación de locutor

Son muchas las aproximaciones que se han realizado al problema de reconocimiento de locutor por lo que sólo se van a comentar las técnicas más empleadas. Éstas se pueden agrupar en tres grupos principales en función de su principio de funcionamiento:

2.4.1 Basadas en la comparación de patrones

Estas técnicas determinan la distancia de similitud entre un fragmento acústico que se supone representativo de la identidad del emisor y un tramo patrón del hablante a reconocer. Destaca la técnica Alineamiento Temporal Dinámico (en inglés Dynamic Time Warping, DTW).

2.4.1.1 Alineamiento Temporal Dinámico (DTW)

Este tipo de técnica surge como respuesta a los problemas derivados de la variabilidad temporal propia del proceso del habla. Cuando se habla, el mismo mensaje se puede generar a distintas velocidades, lo que genera una distorsión temporal importante cuando se desean comparar patrones acústicos.

Estas distorsiones no solo afectan a la duración total del mensaje, influyen en los elementos fonéticos que lo componen. A lo que hay que añadir que las variaciones temporales no son proporcionales a la velocidad de locución y son distintas de un locutor a otro, e incluso para un mismo locutor.

La manera óptima de alinear patrones es utilizar la normalización dinámica (DTW). Para ilustrar el funcionamiento de esta técnica, se supone que se tienen dos patrones $P_1(n)$ y $P_2(m)$, que constan de N y M vectores de parámetros respectivamente. La normalización pretende obtener una función que asocie cada una de las partes de $P_1(n)$ con las correspondientes de $P_2(m)$ de la forma m=f(n). Esta función f(n) debe cumplir las siguientes restricciones:

- f(1)=1
- f(N)=M

Esta función debe relacionar los *n* del eje de tiempo de P_1 con los *m* del eje de tiempo de P_2. Dados dos patrones cualesquiera, f(n) es la función camino que minimiza la distancia entre ellos y se obtiene minimizando la siguiente expresión:

$$D^* = \min\left\{\sum_{n=1}^{N} d\left[P_1(n), P_2(f(n))\right]\right\} \tag{2.3}$$

Siendo $d[P_1(n),P_2(f(n))]$ la distancia entre el instante correspondiente a la parte *n* de P_1 y el instante correspondiente a la parte *f(n)* de P_2 y D la distancia acumulada sobre el camino óptimo. La solución a esta expresión matemática se calcula utilizando algoritmos de programación dinámica.

A continuación podemos ver la representación grafica de la función de alineamiento temporal:

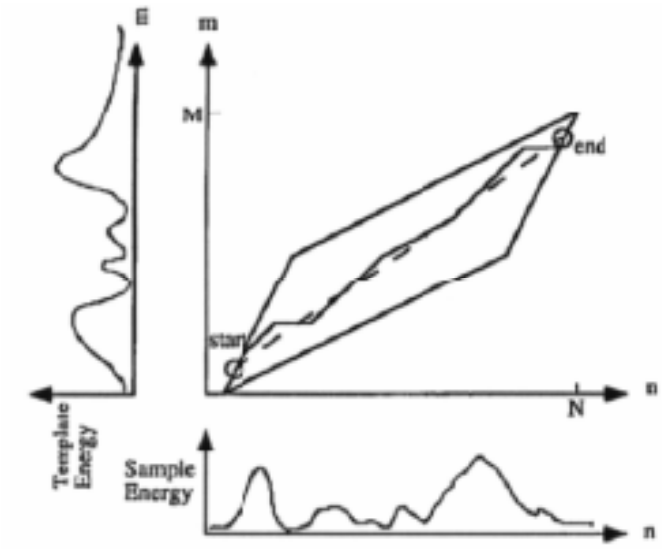


Imagen 2-13. Representación gráfica de la función de Alineamiento Temporal Dinámico (DTW)

2.4.2 Basadas en redes neuronales

La topología de estos modelos de cálculo trata de emular los mecanismos de interconexión que se producen entre las neuronas cerebrales humanas. Se han propuesto diferentes arquitecturas de redes (perceptrón, multicapa, NN de retardo temporal, funciones de base radial, etc.) que conjugadas con ciertas estrategias de agrupación posibilitan diversas técnicas de clasificación como son la del Máximo global o la Binary Tree Search.

2.4.2.1 Redes Neuronales (ANN)

Las redes de neuronas tratan de imitar al tejido nervioso humano encargado de realizar las tareas de percepción sensorial. Pero existe el problema de que no se conoce suficientemente bien el funcionamiento de los tejidos nerviosos que se quieren imitar, por lo que el rendimiento de las redes de neuronas artificiales no es comparable al de las reales.

Las redes neuronales se enmarcan dentro de los procesos distribuidos paralelos. Estos consisten en redes formadas por unidades de cálculo simples interconexionadas, las neuronas, donde las operaciones se realizan de forma distribuida y paralela.

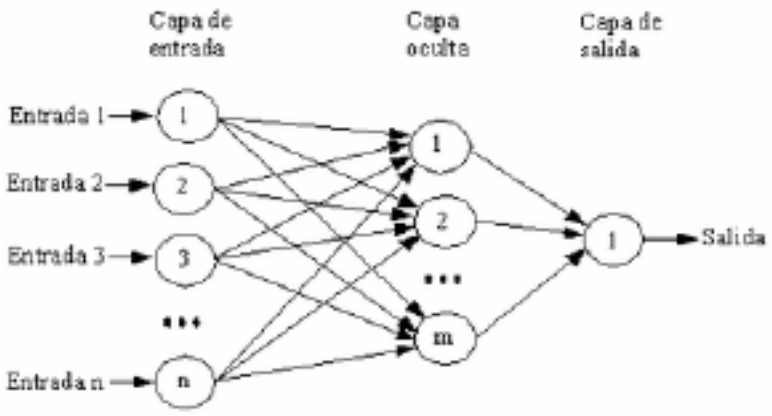


Imagen 2-14. Esquema de una red neuronal de tres capas

Cada neurona se caracteriza por tener un número variable de entradas. Su función es generar a partir de éstas una respuesta (aplicando una función lineal o no).

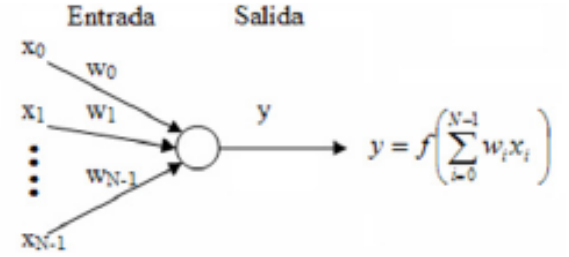


Imagen 2-15. Esquema de una neurona

El inconveniente que aparece es que las arquitecturas paralelas no se programan fácilmente y los procesos distribuidos paralelos dependen de algoritmos automáticos de aprendizaje que resultan bastante complejos.

El empleo de esta técnica en el reconocimiento de locutores ha proporcionado resultados positivos en entornos controlados. Se han propuesto dos metodologías para abordar el problema: la clasificación directa, la red discrimina las características de un locutor frente a otro, y el modelado predictivo del locutor, donde se generan modelos de los locutores. En la primera solución, la incorporación de nuevos usuarios supone el reentrenamiento de la red completa.

2.4.3 Basadas en la modelación de clases fonéticas

En este tipo de técnicas es necesario realizar una clasificación previa de las diversas clases fonéticas para poder modelar las características acústicas dependientes del locutor. Destacan las técnicas siguientes:

- Cuantificadores Vectoriales (VQ)
- Modelos Ocultos de Markov (HMM)
- Modelos de Mezclas de Gaussianas (GMM)

2.4.3.1 Cuantificadores Vectoriales (VQ)

Los vectores de características obtenidos de los locutores muchas veces presentan un tamaño excesivo. La cuantificación vectorial VQ, consiste en obtener un reducido número de vectores relacionados con las características específicas del hablante, de manera que los elementos de cada vector representen las propiedades de un hablante específico.

A cada vector se le denomina *codework* o *centroide* y al conjunto de éstos se le denomina *codebook*. El tamaño del *codebook* influye directamente sobre las prestaciones y la carga computacional del sistema.

Para llevar a cabo la verificación de locutor, se procesa la locución calculando la distancia acumulada entre los vectores de parámetros y el *codebook* asociado a ese locutor. Finalmente, se aceptará o no al locutor comparando esta distancia con un umbral determinado.

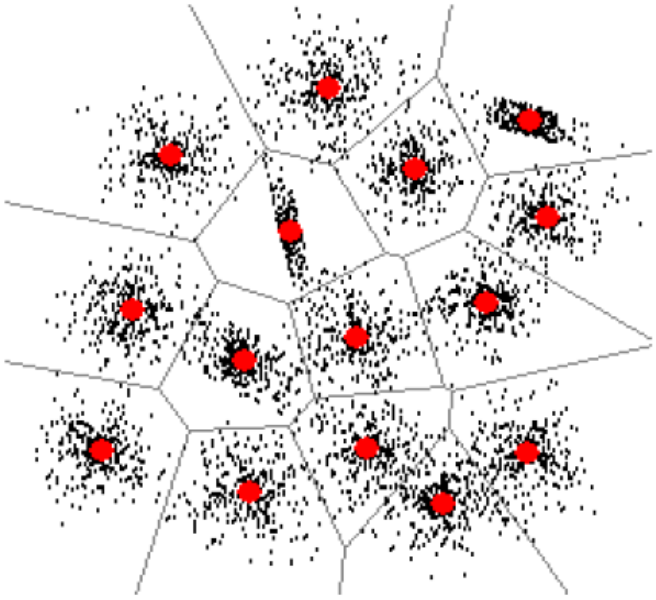


Imagen 2-16. Esquema de espacio bidimensional mapeado con codework

El uso de esta técnica supone una reducción de la complejidad computacional en el cálculo de distancias (se pueden usar cálculos tan simples como la distancia euclídea o la de Mahalanobis). Su principal inconveniente es la distorsión introducida por el error de cuantificación al asociar a cada vector un representante.

2.4.3.2 Modelos Ocultos de Markov (HMM)

Una técnica de modelado estocástico muy utilizada son los modelos ocultos de Markov (Hidden Markov Models, HMM en sus siglas en inglés). La teoría básica de los modelos ocultos de Markov fue introducida por primera vez por Baum y sus colaboradores en una serie de

artículos entre los años 1966 y 1972. La comparación de patrones en reconocimiento de locutor dependiente de texto se realiza midiendo la verosimilitud de una observación con el modelo de locutor que dice ser.

Un HMM es una máquina de estado finita, en la que las observaciones son una función probabilística del estado, es decir, el modelo es un proceso doblemente estocástico formado por un proceso estocástico oculto no observable directamente, que corresponde a las transiciones entre estados y un proceso estocástico observable cuya salida es la secuencia de vectores espectrales.

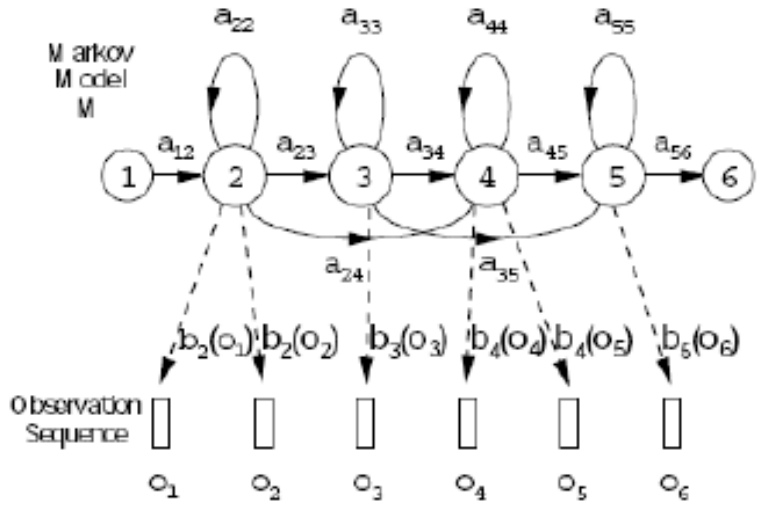


Imagen 2-17. El modelo de generación de Markov

En la Imagen 2-17 se pueden observar los elementos que definen un HMM:

- N: Representa el número de estados del modelo, donde q_t denota el estado en el instante de tiempo t.

$$S = \{s_1, s_2, \ldots, s_N\}$$

- M: Representa la dimensión del conjunto de observaciones distintas de salida, es decir, el tamaño del alfabeto.

$$V = \{v_1, v_2, \ldots, v_M\}$$

- La distribución de probabilidad de transición entre estados $A=\{a_{ij}\}$:

$$a_{ij} = P(q_t=s_j \mid q_{t-1}=s_i) \quad 1 \le i, j \le N$$

- La distribución de probabilidad de emisión de símbolos entre estados $B=\{b_j(v_k)\}$.

$$b_j(v_k) = P(o_t=v_k \mid q_t=s_j) \quad 1 \le j \le N \quad 1 \le k \le M \quad v_k \in V$$

- Distribuciones del estado inicial $\Pi = \{\pi_i\}$.

$$\pi_i = P(q_0=s_i) \quad 1 \le i \le N$$

Con todo esto un HMM se puede describir como $\lambda = \{A, B, \Pi\}$. A partir de ahora vamos a tratar de resolver los tres principales problemas que existen para que un HMM tenga utilidad en aplicaciones reales, estos son:

- Evaluación de la probabilidad.
- Encontrar la secuencia de estados óptima.
- Entrenamiento de un modelo.

2.4.3.2.1 Evaluación de la probabilidad

Dada una secuencia de observación O={o_1, o_2,…, o_T} y un modelo λ = {A, B, Π}, ¿cómo calculamos P(O|λ), la probabilidad de la secuencia de observación? Si es posible calcular esta probabilidad, entonces se podría calcular para todos los modelos y escoger aquel para el cual la probabilidad sea mayor. Existen tres métodos diferentes para evaluar la probabilidad:

- Manera directa.
- Algoritmo de avance.
- Algoritmo de retroceso.

– *Manera directa*

La manera más directa de solucionarlo sería enumerando todas las posibles secuencias de estados de longitud T que generen la secuencia de observación O y sumando sus probabilidades según el teorema de la Probabilidad Total:

$$P(O \mid \lambda) = \sum_{Q} P(O \mid Q, \lambda) \cdot P(Q \mid \lambda) \tag{2.4}$$

Para ello consideremos determinada una secuencia de estados Q=(q_1, q_2,…,q_T) donde q_1 es el estado inicial. La probabilidad de la secuencia de observación O dada la secuencia de estados Q (asumiendo independencia estadística de las observaciones) es:

$$P(O \mid Q, \lambda) = \prod_{t=1}^{T} P(o_t \mid q_t, \lambda) \tag{2.5}$$

Por lo tanto se obtiene:

$$P(O \mid Q, \lambda) = b_{q_1}(o_1) \cdot b_{q_2}(o_2) \cdots b_{q_T}(o_T) \tag{2.6}$$

Por otra parte la probabilidad de la secuencia Q se puede expresar como:

$$P(Q \mid \lambda) = \pi_{q_1} \cdot a_{q_1 q_2} \cdot a_{q_2 q_3} \cdots a_{q_{T-1} q_T}$$

Esta probabilidad se interpreta como la probabilidad del estado inicial, multiplicada por las probabilidades de transición de un estado a otro. Sustituyendo las ecuaciones (2.5) y (2.6) en la ecuación (2.4) se obtiene la probabilidad de la secuencia de observación:

$$\boxed{P(O \mid \lambda) = \sum_{Q} P(O \mid Q, \lambda) \cdot P(Q \mid \lambda) = \sum_{q_1, q_2, \ldots, q_T} \pi_{q_1} \cdot b_{q_1}(o_1) \cdot a_{q_1 q_2} \cdot b_{q_2}(o_2) \cdots a_{q_{T-1} q_T} \cdot b_{q_T}(o_T)} \tag{2.7}$$

La interpretación de este resultado es bien sencilla: inicialmente en el tiempo t=1 nos encontramos en el estado q_1 con probabilidad π_{q1} y generamos el símbolo o_1 con probabilidad $b_{q1}(o_1)$. Al avanzar el reloj al instante t=2 se produce una transición al estado q_2 con probabilidad a_{q1q2} y generamos el símbolo o_2 con probabilidad $b_{q2}(o_2)$. Este proceso se repite hasta que se produce la ultima transición del estado q_{T-1} al estado q_T con probabilidad a_{qT-1qT} y generamos el símbolo o_T con probabilidad $b_{qT}(o_T)$.

A pesar de haber llegado al resultado deseado se puede ver fácilmente que no es una manera muy eficiente de calcular la probabilidad ya que requiere utilizar $2T \cdot N^T$ operaciones, por lo que su complejidad es del orden **$O(N^T)$** y resulta computacionalmente intratable.

- *Algoritmo de avance (Forward Algorithm)*

Afortunadamente existe una manera más eficiente de llegar al mismo resultado. La clave está en guardar los resultados intermedios y utilizarlos para los posteriores cálculos de la secuencia de estados. A este algoritmo se le denomina el Algoritmo de Avance.

El primer paso es definir la variable hacia delante de la siguiente manera:

$$\alpha_t(i) = P(o_1, o_2, \ldots, o_t, q_t = s_i \mid \lambda) \tag{2.8}$$

Esta variable corresponde con la probabilidad de que el modelo λ se encuentre en el estado s_i habiendo generado la secuencia parcial o_1, o_2,…, o_t hasta el instante de tiempo t. $\alpha_t(i)$ se puede calcular por inducción siguiendo los siguientes pasos:

1. *Inicialización*:
$$\alpha_1(i) = \pi_i \cdot b_i(o_1) \qquad 1 \le i \le N \tag{2.9}$$
En este proceso se inicializan tanto las probabilidades hacia adelante como la probabilidad conjunta del estado s_i y la observación o_t.
2. *Inducción*:
$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) \cdot a_{ij}\right] \cdot b_j(o_{t+1}), \qquad 1 \le t \le T-1 \quad 1 \le j \le N \tag{2.10}$$
La expresión entre corchetes representa la probabilidad de alcanzar el estado s_j en el instante de tiempo t+1 partiendo de todos los estados posibles s_i en el instante t y observando la secuencia parcial $o_1,o_2,\ldots,o_t$ hasta el instante t. Si multiplicamos ahora, desde el estado s_j, dicho término por la probabilidad de observar o_{t+1} obtenemos $\alpha_{t+1}(j)$.
3. *Finalización*:
$$\boxed{P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i)} \tag{2.11}$$
El cálculo final se realiza sumando todas las variables hacia adelante $\alpha_T(i)$ en el instante final T. Por definición $\alpha_T(i)$ es la probabilidad conjunta de observar la secuencia completa O y encontrarnos en el estado final s_i: $\alpha_T(i)=P(o_1, o_2,\ldots, o_T, q_T=s_i|\lambda)$, con lo que si sumamos dicha probabilidad para todos los estados posibles obtenemos la probabilidad esperada $P(O|\lambda)$.

- *Algoritmo de retroceso (Backward Algorithm)*

De forma sintética podemos calcular la variable hacia atrás de manera similar a como se definió la probabilidad hacia adelante:

$$\beta_t(i) = P(o_{t+1}, o_{t+2}, \ldots, o_T \mid q_t = s_i, \lambda) \tag{2.12}$$

Esta probabilidad $\beta_t(i)$ es en este caso la probabilidad de generar la observación parcial $O=\{o_{t+1}, o_{t+2},\ldots, o_T\}$ desde el instante t+1 hasta el instante final T dado que el modelo se encuentra en el estado s_i en el instante de tiempo t. $\beta_t(i)$ se puede calcular por inducción como sigue:

1. *Inicialización*:
$$\beta_T(i) = 1 \qquad 1 \le i \le N \tag{2.13}$$

2. *Inducción*:

$$\beta_t(i) = \sum_{i=1}^{N} a_{ij} \cdot b_j(o_{t+1}) \cdot \beta_{t+1}(j), \quad t = T-1, T-2, \ldots, 1 \quad 1 \le i \le N \qquad (2.14)$$

Una vez que tenemos la probabilidad desde el estado s_j de terminar la secuencia desde o_{t+2} gracias a $\beta_{t+1}(j)$ sólo necesitamos multiplicar por la probabilidad de observar o_{t+1} desde el mismo estado s_j por la probabilidad de la transición de venir del estado s_i al s_j. Si sumamos para todos los posibles estados s_i tendremos completa la observación desde cualquier estado s_i a partir de o_{t+1} que es exactamente la definición de β_t (i).

3. *Finalización*:

$$P(O \mid \lambda) = \sum_{i=1}^{N} \pi_i \cdot b_i(o_1) \cdot \beta_1(i) \qquad (2.15)$$

Con $\beta_1(i)$ tenemos la probabilidad desde un estado s_i de observar la secuencia O a partir de o_2. Solamente necesitamos multiplicar por la probabilidad de observar o_1 desde ese estado s_i (determinada por $b_i(o_1)$) y por la probabilidad de estar en s_i inicialmente (dada por π_i). Una vez hecho esto tendremos la secuencia completa observada y solo nos queda sumar para cada posible estado s_i inicial.

Los últimos algoritmos se pueden combinar[6] para calcular P(O|λ):

$$P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_t(i) \cdot \beta_t(i) \qquad (2.16)$$

La complejidad de estos dos algoritmos (el de avance y el de retroceso) comparada con la de la manera directa de calcular P(O|λ) es mucho menor y se encuentra en ambos casos en el orden **$O(N^2T)$**, con lo que el ahorro computacional es claro.

Algoritmo	Complejidad
Manera directa	$O(N^T)$
Algoritmo de avance	$O(N^2T)$
Algoritmo de retroceso	$O(N^2T)$

Tabla 2-1. Comparación de la complejidad de los 3 algoritmos para evaluar la probabilidad

2.4.3.2.2 Encontrar la secuencia de estados óptima

Decodificar un HMM consiste en encontrar la secuencia de estados óptima, dada una secuencia de observación. La resolución de este problema resulta muy importante para tareas de segmentación y reconocimiento de voz.

A diferencia del apartado 2.4.3.2.1 para el que se suele dar una solución exacta, existen diferentes maneras de resolver este problema. La razón es que la definición de secuencia óptima no es única, sino que existen varios criterios de optimización.

El criterio más utilizado es el algoritmo de Viterbi, que trata de encontrar la mejor secuencia de estados, es decir, maximizar la probabilidad P(Q|O, λ) o lo que es equivalente, maximizar P(Q, O|λ). En la práctica, también se puede utilizar para evaluar HMMs.

[6] La combinación de ambos algoritmos se denomina algoritmo Forward-Backward o Avance-Retroceso en español.

Para encontrar la mejor secuencia de estados Q={q_1, q_2,…, q_T} para una secuencia de observación dada O={o_1, o_2,…, o_T} se define la variable $\delta_t(i)$:

$$\delta_t(i) = \max_{q_1,q_2,\ldots q_{t-1}} P\left[q_1, q_2, \ldots q_t = s_i, o_1, o_2, \ldots o_t \mid \lambda\right] \tag{2.17}$$

Esta variable representa la secuencia de estado con mayor probabilidad en el instante t que acaba en el estado s_i y que ha generado las t primeras observaciones.

A continuación se sigue un algoritmo de inducción similar al anterior, con la excepción de que en vez de tomar la suma de las probabilidades de los diferentes caminos que acaban en un mismo estado, el algoritmo de Viterbi selecciona y recuerda el mejor camino.

1. *Inicialización*:

$$\delta_1(\mathrm{i}) = \pi_\mathrm{i} \cdot \mathrm{b_i}(\mathrm{o}_1) \qquad 1 \le \mathrm{i} \le \mathrm{N} \tag{2.18}$$

$$\phi_1(\mathrm{i}) = 0 \tag{2.19}$$

   Se define $\delta_1(i)$ como la probabilidad de estar en el estado s_i en el instante t=1 por la de generar el símbolo o_1.

   En el vector ϕ se va a almacenar el argumento que maximiza $\delta_t(i)$ para cada valor de t y de j; toma inicialmente el valor 0.

2. *Recursión*:

$$\delta_t(j) = \max_{1 \le i \le N}\left[\delta_{t-1}(i) \cdot a_{ij}\right] \cdot b_j(o_t), \qquad 2 \le \mathrm{t} \le \mathrm{T} \quad 1 \le \mathrm{j} \le \mathrm{N} \tag{2.20}$$

$$\phi_t(j) = \arg\max_{1 \le i \le N}\left[\delta_{t-1}(i) \cdot a_{ij}\right], \qquad 2 \le \mathrm{t} \le \mathrm{T} \quad 1 \le \mathrm{j} \le \mathrm{N} \tag{2.21}$$

3. *Finalización*:

$$P^* = \max_{1 \le i \le N}\left[\delta_t(i)\right] \tag{2.22}$$

$$q_t^* = \arg\max_{1 \le i \le N}\left[\delta_t(i)\right] \tag{2.23}$$

4. *Backtracking*:

$$q_t^* = \varphi_{t+1}\left(q_{t+1}^*\right), \qquad \mathrm{t} = \mathrm{T}-1, \mathrm{T}-2, \ldots, 1 \tag{2.24}$$

   En este último paso se reconstruye la secuencia de estados partiendo desde el estado final hasta llegar al principio.

2.4.3.2.3 Entrenamiento de un modelo

Nos encontramos ante el más complicado de los tres problemas que anunciábamos anteriormente. Este problema plantea cómo se deben ajustar los parámetros del modelo {A, B, Π} para maximizar la probabilidad de observación dado el modelo P(O|λ).

El principal inconveniente es que no existe ningún método analítico conocido que maximice el conjunto de parámetros a partir de los datos de entrenamiento. Se puede resolver, sin embargo utilizando un algoritmo iterativo como el algoritmo de Baum-Welch. Este algoritmo utiliza los mismos principios que el algoritmo EM (Expectation Maximization).

Este algoritmo es un método de optimización local por lo que requiere de parámetros {A, B, Π} iniciales y tiene el problema de que puede no llegar a la mejor solución debido a los mínimos locales. El procedimiento consiste en actualizar los pesos de forma iterativa para poder explicar mejor las secuencias de entrenamiento observadas.

Este método procede a la estimación de {A, B, Π} que denotamos como $\{\hat{A}, \hat{B}, \hat{\Pi}\}$, apoyándose en:

- $\hat{\pi}_i = \text{número esperado de veces de estar en } s_i \text{ en } t = 1$
- $\hat{a}_{ij} = \dfrac{\text{número esperado de transiciones realizadas desde } s_i \text{ hasta } s_j}{\text{número esperado de transiciones realizadas desde } s_i}$
- $\hat{b}_j(v_k) = \dfrac{\text{número esperado de veces de estar en } s_j \text{ y observar el símbolo } v_k}{\text{número esperado de veces de estar en } s_j}$

La relación entre α y β adyacentes se puede observar mejor en la siguiente figura. α se calcula recursivamente de izquierda a derecha mientras β se calcula recursivamente de derecha a izquierda.

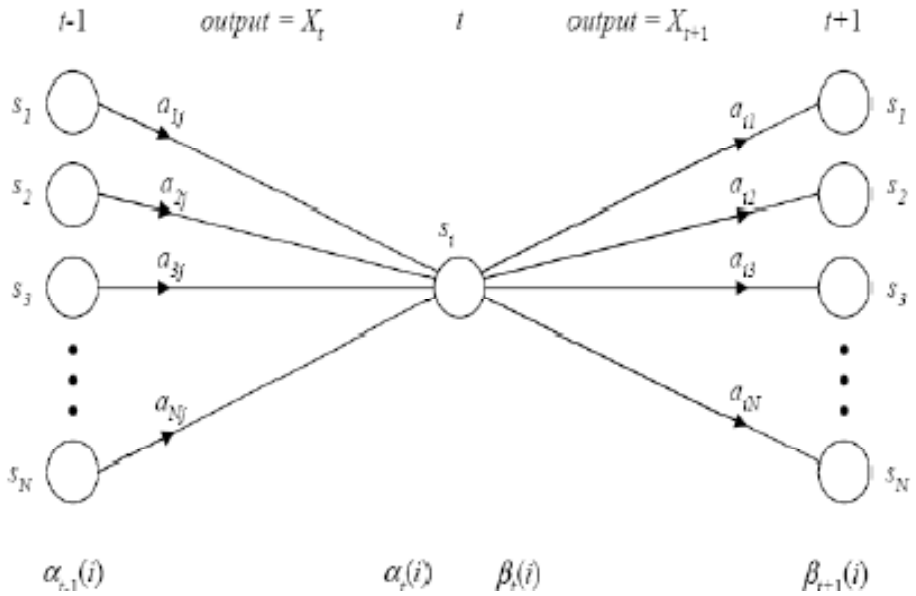


Imagen 2-18. Relación entre α y β en el algoritmo Forward-Backward

A continuación definimos la variable $\varepsilon_t(i,j)$ que representa la probabilidad de realizar una transición del estado s_i al estado s_j en el instante de tiempo t dado el modelo y dada la secuencia de observación, es decir:

$$\varepsilon_t(i,j) = P\left(q_t = s_i, q_{t+1} = s_j \mid O, \lambda\right) = \frac{P\left(q_t = s_i, q_{t+1} = s_j, O \mid \lambda\right)}{P\left(O \mid \lambda\right)} = \frac{\alpha_t(i) \cdot a_{ij} \cdot b_j\left(o_{t+1}\right) \cdot \beta_{t+1}(j)}{\sum_{k=1}^{N} \alpha_T(k)} \quad (2.25)$$

Este resultado se puede ilustrar mejor en la siguiente figura:

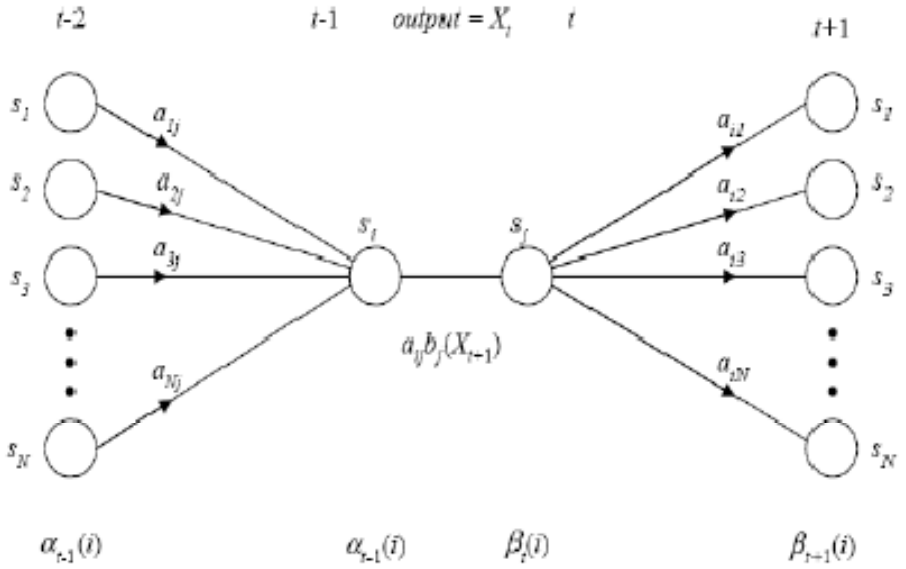


Imagen 2-19. Operaciones para el cálculo de $\varepsilon_t(i,j)$

También es necesario calcular: $$\gamma_t(i) = \sum_{j=1}^{N} \varepsilon_t(i,j) \tag{2.26}$$

que, al sumar para cada estado de destino, representa la probabilidad de realizar una transición desde el estado s_i hacia cualquier estado en el instante de tiempo t.

Con todo esto podemos deducir:

- Número de transiciones desde el estado s_i: $$\sum_{t=1}^{T-1} \gamma_t(i) \tag{2.27}$$
- Número de transiciones del estado s_i al s_j: $$\sum_{t=1}^{T-1} \varepsilon_t(i,j) \tag{2.28}$$

Dados estos resultados ya podemos estimar los nuevos parámetros del modelo:

$$\boxed{\hat{\pi}_i = \gamma_1(i) \qquad 1 \le i \le N} \tag{2.29}$$

$$\boxed{\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \varepsilon_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \qquad 1 \le i, j \le N} \tag{2.30}$$

$$\boxed{\hat{b}_j(v_k) = \frac{\sum_{t \in o_t = v_k} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \quad 1 \le j \le N \qquad 1 \le k \le M} \tag{2.31}$$

De acuerdo con el algoritmo EM, el algoritmo Baum-Welch garantiza una mejora monótona en la probabilidad en cada iteración hasta que ésta converge en un máximo local. El algoritmo se puede resumir en los siguientes pasos:

1. **Inicialización**: Se escoge una estimación inicial del modelo λ.
2. **Paso E**: se calcula la variable $\varepsilon_t(i, j)$ a partir de λ.
3. **Paso M**: se calcula $\hat{\lambda}$ con las ecuaciones de reestimación.
4. **Iteración**: λ pasa a tomar el valor de $\hat{\lambda}$ y se repite el algoritmo desde el paso 2 hasta que converge.

2.4.3.3 Modelos de Mezclas de Gaussianas (GMM)

Los GMMs modelan la señal de voz como una suma finita de gaussianas. Son usados para reconocimiento de hablante como un modelo probabilístico de densidades multivariables capaz de representar densidades arbitrarias, las cuales se emplean en aplicaciones independientes del texto.

Este tipo de técnicas proporciona unos resultados muy buenos con una complejidad menor que otro tipo de técnicas como los HMMs. Aunque presenta la necesidad de tener que emplear gran cantidad de datos para entrenar los modelos de reconocimiento.

El verificador implementado en este proyecto se encuentra situado dentro del grupo de verificadores basados en la modelación de clases fonéticas y la técnica empleada ha sido utilizar Modelos de Mezclas Gaussianas (GMM). De modo que este punto lo veremos con más detalle en el siguiente capítulo.

CAPÍTULO III

Descripción del sistema

3.1 Verificador de Locutor

Como ya se ha comentado anteriormente, en este proyecto se ha implementado un Verificador de Locutor independiente de texto basado en Modelos de Mezclas de Gaussianas (GMM). Se puede considerar que los modelos GMM son una particularización de los HMM de un estado, modelado por un conjunto de Gaussianas.

Generalmente suele hablarse de un sistema GMM-UBM en el que las siglas UBM (Universal Background Model o Modelo Universal de Mundo) hacen referencia al modelo de mundo usado para hacer la verificación. En este punto, cabe recordar que el funcionamiento de un verificador consiste en decidir si la muestra de voz es más verosímil que fuera generada por el usuario indicado o por el modelo de mundo.

Como ya se ha comentado en alguna otra ocasión durante este trabajo, el uso de GMM se considera mejor para aplicaciones en las que el verificador sea independiente del texto. Un sistema GMM permite obtener unas prestaciones comparables o superiores al de técnicas más complejas como HMM con el consiguiente ahorro computacional.

Los sistemas de verificación de locutor independiente de texto basados en GMM-UBM se caracterizan por poseer una base de datos de modelos de locutor y un modelo de mundo. Su funcionamiento consiste en que un locutor no identificado afirma ser uno de los locutores de la base de datos y el sistema debe verificar si este locutor es quien dice ser. Para ello debe ser capaz de extraer las características de la señal de voz proporcionada por el locutor que desea verificarse, como vimos en el capítulo anterior, y de tomar una decisión.

A continuación vamos a explicar con más detalle las tres fases esenciales de este tipo de verificadores:

- Fase de entrenamiento.
- Fase de verificación.
- Fase de test.

3.1.1 Fase de entrenamiento

El objetivo de la fase de entrenamiento es la obtención de los modelos necesarios para realizar la verificación de locutor. Se debe generar un modelo para cada usuario y el modelo de mundo con el que se realizará la comparación en la fase de verificación.

Entradas:	Base de Datos con todos los ficheros de parámetros de cada usuario.
Salidas:	Modelo de Mundo Modelos personales de cada usuario de la BBDD.

Tabla 3-1. Entradas y salidas de la Fase de Entrenamiento

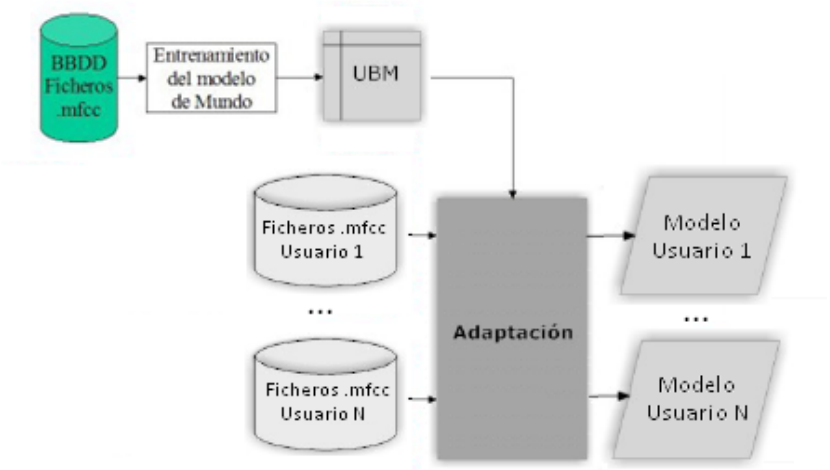


Imagen 3-1. Fase de entrenamiento del verificador

En este punto podemos explicar la necesidad de crear un módulo grabador en la PDA que se encargará de mantener una base de datos con todas las locuciones de entrenamiento de los diferentes usuarios. De la misma manera, también comprobamos la utilidad del siguiente módulo de Parametrización que obtiene las principales características de dichas locuciones creando una nueva base de datos de ficheros parametrizados. La utilización de estos dos módulos es un proceso previo a la fase de entrenamiento de nuestro verificador y procederemos a explicarla más adelante.

Dentro de la fase de entrenamiento observamos que para obtener el modelo de cada usuario hace falta utilizar el último módulo que faltaba por comentar, el Adaptador que también se explicará con detalle más adelante.

3.1.1.1 Modelos de Mezclas de Gaussianas

Un paso importante para implementar el sistema que se está tratando, es elegir la función de densidad $p(X|\lambda)$[7]. La selección depende de las especificaciones de la aplicación para la que está destinado el sistema. En el caso de este proyecto, verificación de locutor independiente de texto, en el que no se sabe qué puede decir el interlocutor, la función que más se adapta es la mezcla de Gaussianas.

Si se tiene un vector de características x de dimensión D, la densidad de probabilidad $p(x|\lambda)$ es una combinación (o mezcla) de M Gaussianas simples $p_i(x)$ ponderadas por los coeficientes (o pesos) w_i.

$$p(x|\lambda) = \sum_{i=1}^{M} w_i \cdot p_i(x) \tag{3.1}$$

Los pesos w_i tienen que cumplir la siguiente restricción:

$$\sum_{i=1}^{M} w_i = 1 \tag{3.2}$$

y las Gaussianas simples $p_i(x)$ tienen cada una un vector de medias μ_i de dimensiones 1×D y una matriz D×D de covarianzas, Σ_i; se definen como:

$$p_i(x) = \frac{1}{\sqrt{(2\pi)^D |\Sigma_i|}} e^{-\frac{1}{2}(x-\mu_i)'(\Sigma_i)^{-1}(x-\mu_i)} \tag{3.3}$$

El modelo se denota con todos sus parámetros como $\lambda = \{w_i, \mu_i, \Sigma_i\}$ con i = 1,…, M.

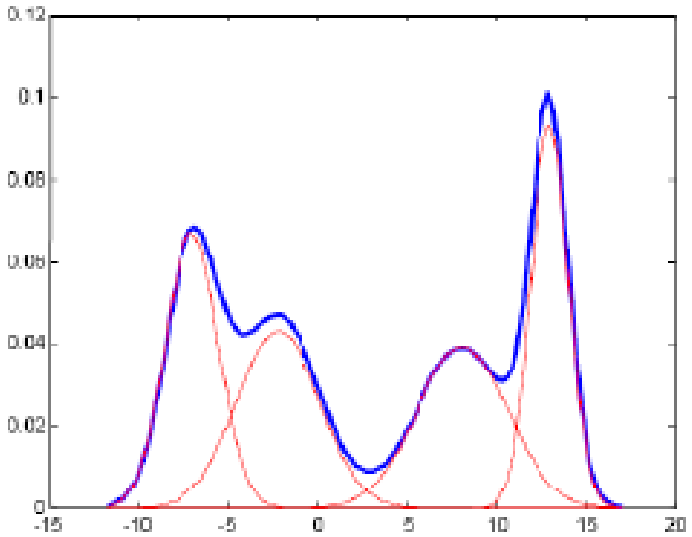


Imagen 3-2. Ejemplo de un GMM con cuatro Gaussianas

Se aprecia entonces que los GMM son modelos de densidad semiparamétricos, pues cuentan con una serie de parámetros que hay que estimar asumiendo estas formas Gaussianas y, sin embargo, se cuenta con varios grados de libertad que permiten un modelado arbitrario de la función de densidad de probabilidad sin que conlleve mucho gasto computacional. Está, por lo tanto, a caballo entre los modelos paramétricos y no paramétricos.

[7] $p(X|\lambda)$ representa la probabilidad de que una secuencia de entrada X pertenezca al modelo definido por λ.

3.1.1.1.1 Modelo de mundo

Esta idea se plasma en los sistemas UBM en la generación de un modelo de mundo o modelo de no usuario. El modelo UBM es un GMM entrenado para representar la distribución de características independientes del hablante.

A la hora de entrenar el UBM hay que tener en cuenta que no existen medidas objetivas que determinen el número óptimo de hablantes que se debe emplear ni la duración de las locuciones. Una consideración que sí debe ser tenida en cuenta es la población que se quiere que represente el modelo. Por ejemplo, se puede desear un UBM para mujeres y otro para hombres o utilizar un UBM que represente a toda la población. Si se entrena un único modelo UBM se debe prestar cuidado en que los datos empleados para el entrenamiento estén balanceados para las diferentes subpoblaciones.

Una vez recopilados todos los datos que se van a emplear para el entrenamiento del UBM, se pueden seguir dos estrategias diferentes para obtenerlo.

La primera de ellas consiste en entrenar un único modelo UBM a partir del conjunto completo de las locuciones de los usuarios. En este caso se debe tener en cuenta lo comentado con anterioridad acerca del balance de los datos para distintas subpoblaciones.

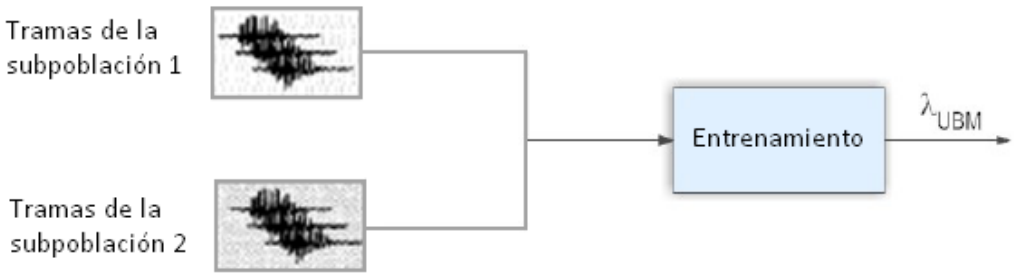


Imagen 3-3. Información de diferentes subpoblaciones se une antes de entrenar el UBM

La otra forma de entrenar el modelo UBM consiste en entrenar un modelo para cada una de las subpoblaciones consideradas, hombres/mujeres, distintos micrófonos empleados, etc. Una vez obtenidos, el modelo general se obtiene como combinación de éstos. Este método permite emplear datos no balanceados y controlar la composición final del modelo UBM.

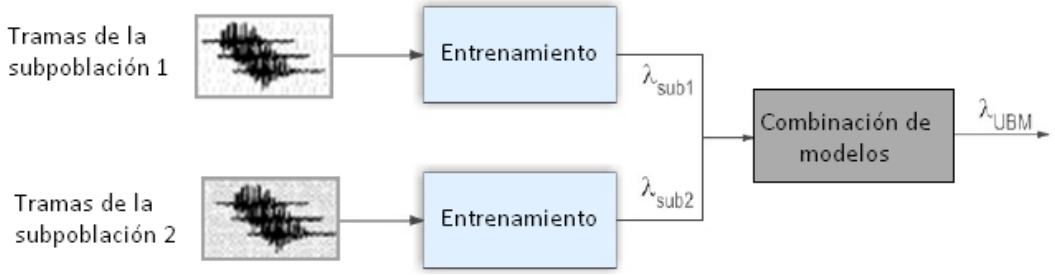


Imagen 3-4. Se entrenan por separado 2 modelos de diferentes subpoblaciones y luego se combinan para dar lugar al UBM

- *Algoritmo EM para un GMM.*

  En el entrenamiento de los modelos se emplea el algoritmo EM (como vimos en el caso de los HMM) que proporciona una técnica iterativa para estimar la máxima verosimilitud de un conjunto de parámetros, en los que existen datos ocultos dependientes estadísticamente de los datos a estimar y de los datos observados.

Cada modelo está formado por una composición lineal ponderada de gaussianas. Los parámetros que definen cada modelo son:

- Vector de medias: $\mu_p = \{\mu_{ip}\}$
- Matriz de covarianzas: $\Sigma_p = \{\Sigma_{ip}\}$
- Vector de pesos: $\omega_p = \{\omega_{ip}\}$

De forma que los modelos vienen definidos por la terna de parámetros $\lambda_p = \{\mu_{ip}, \Sigma_{ip}, \omega_{ip}\}$.

El algoritmo EM [REY95, DEM77, MCL97, LIT02] es un procedimiento paramétrico iterativo diseñado para obtener estimaciones de máxima verosimilitud cuando una muestra de datos es parcialmente observada.

Sea Y = (Y_{obs}, Y_{per}) una muestra *multivariable*[8] incompleta, donde Y_{obs} *y* Y_{per} son la parte observada y no observada de *Y*, respectivamente; θ el vector de parámetros desconocidos de la distribución de probabilidad *P* para los datos completos; y L(θ|Y) la función de log-verosimilitud asociada. En la *k*-ésima iteración del algoritmo EM se ejecutan dos pasos:

1. **Paso E.** Dada una estimación $\theta^{(k)}$ de θ, calcular $Q\left(\theta;\theta^{(k)}\right)$, donde:
$$Q\left(\theta;\theta^{(k)}\right) = \int \ln L\left(\theta \mid Y\right) \cdot P\left[Y_{per} \mid Y_{obs}, \theta^{(k)}\right] \cdot dY_{per} \qquad (3.4)$$
2. **Paso M.** Encontrar $\theta^{(k+1)}$ tal que $\theta^{(k+1)} = \arg\max_{\theta} Q\left(\theta;\theta^{(k)}\right)$.

Partiendo de un punto inicial $\theta^{(0)}$, los pasos E y M se repiten alternativamente, con lo que se genera una sucesión de estimadores $\left\{\theta^{(k)}\right\}$. El algoritmo EM tiene la propiedad de que en cada iteración se incrementa L(θ|Y), la función de log-verosimilitud para datos observados, así que $L(\theta^{(k+1)}|Y_{obs}) \geq L(\theta^{(k)}|Y_{obs})$, dándose la igualdad si y sólo si $Q\left(\theta^{(k+1)};\theta^{(k)}\right) = Q\left(\theta^{(k)};\theta^{(k)}\right)$. Bajo ciertas condiciones de regularidad, $\left\{\theta^{(k)}\right\}$ converge hacia la estimación de máxima verosimilitud de θ. Para más detalles sobre los aspectos teóricos y prácticos del algoritmo EM se remite al lector a [MCL97] y [LIT02].

Resumiendo, el algoritmo EM ajusta iterativamente los parámetros *λ* del GMM para incrementar monótonamente la probabilidad del modelo estimado para los vectores de características. En este caso, para las iteraciones *k* y *k* + 1, se tiene que $p\left(X \mid \lambda^{(k+1)}\right) \geq p\left(X \mid \lambda^{(k)}\right)$. Para más información sobre las ecuaciones del EM para entrenamiento de GMM ver [REY95].

Desde el momento en el que se asumen independientes los vectores de $X = \{x_1, ..., x_T\}$, la log-probabilidad de un modelo *λ*, para la secuencia de vectores de características *X*, se calcula como:

$$\log p\left(X \mid \lambda\right) = \sum_{t=1}^{T} \log p\left(x_t \mid \lambda\right) \qquad (3.5)$$

con $p(x_t/\lambda)$ calculado según la ecuación (3.3). A menudo se utiliza el valor medio, dividiendo la suma por *T*, para eliminar los efectos de la longitud de la secuencia de voz.

[8] *Multivariable* en el sentido de que hay varias variables medidas para cada individuo u objeto estudiado.

Este factor se puede considerar también una compensación, por asumir independencia cuando realmente no existe.

Un GMM se puede ver como un HMM de un único estado (además de los dos estados virtuales de entrada y salida) con una densidad de observación de mezcla de Gaussianas [REY92]. Será esta observación la que se utilice para aprovechar la herramienta HTK[9] que trabaja con HMM, y por ello con GMM.

Las ventajas de utilizar un GMM como función de probabilidad es que es computacionalmente barato, se basa en un modelo estadístico de comprensión sencilla y no depende de los aspectos temporales de la voz (en términos de independencia del texto). Esto último se puede considerar una propiedad desfavorable, pues se pierden las características temporales de alto nivel que se podrían llegar a aprovechar, pero se puede ver también como una simplificación que no supone pérdidas de características importantes.

3.1.1.1.2 Modelo de usuario

Una vez se ha obtenido el modelo UBM, se obtienen los modelos de cada uno de los locutores como una adaptación del modelo UBM a las muestras de cada locutor. El método empleado para la adaptación se conoce como aprendizaje bayesiano o máximo a posteriori (MAP).

Esta adaptación provoca un mayor acoplamiento entre el modelo UBM y los modelos de los locutores, lo cual produce mejores resultados que los modelos desacoplados y permite emplear técnicas para agilizar el cálculo de la verosimilitud en la fase de test.

En el proceso de adaptación del modelo de mundo, no todas las Gaussianas son adaptadas. Por tanto es posible crear un modelo simplificado que solamente guarde los valores relativos a aquellas Gaussianas que han sufrido modificaciones.

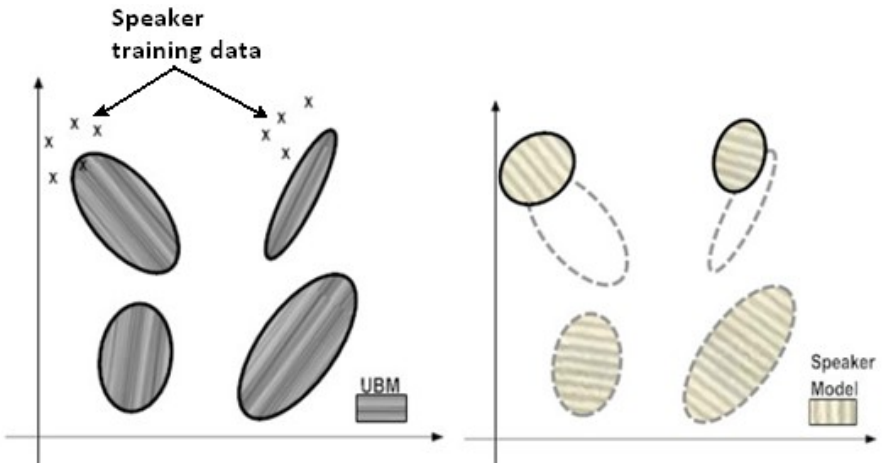


Imagen 3-5. Proceso de adaptación a un locutor del UBM

Otra simplificación que se puede realizar en la búsqueda de disminuir la carga computacional de la fase de decisión, es considerar que del conjunto de Gaussianas sólo unas pocas contribuyen de forma significativa en el cálculo de la diferencia de verosimilitudes. Por lo tanto se pueden computar únicamente las C Gaussianas más relevantes, reduciéndose el número de Gaussianas a evaluar a 2C.

[9] *HTK* (Espacio de trabajo para Modelos Ocultos de Markov del inglés Hidden Markov Model Toolkit) es un conjunto de herramientas para construir y trabajar con HMM.
Portal: *http://htk.eng.cam.ac.uk/*

3.1.2 Fase de verificación

El objetivo de la fase de verificación es dar una respuesta al usuario comprobando si la locución de entrada ha sido producida por el usuario cuyo modelo se ha utilizado en la verificación.

Entradas:	Locución del usuario Modelo del usuario Modelo de mundo Umbral de decisión
Salidas:	Resultado de la verificación

Tabla 3-2. Entradas y salidas de la Fase de Verificación

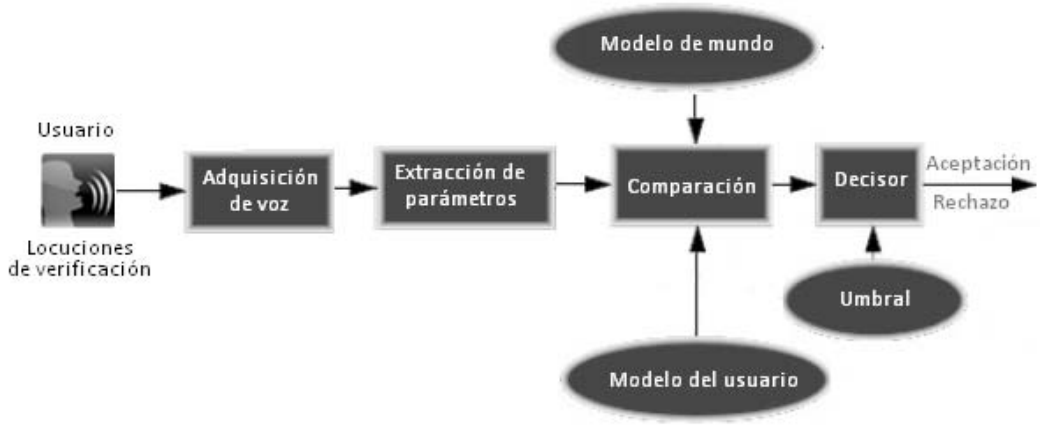


Imagen 3-6. Fase de verificación

Recordamos que la fase de verificación puede ser vista como una decisión binaria entre las siguientes hipótesis:

H_0: X fue pronunciado por el locutor S.

H_1: X no fue pronunciado por el locutor S.

El test óptimo para decidir entre las dos hipótesis es el cálculo de la verosimilitud de que la muestra de voz haya sido generada por cada uno de los dos modelos:

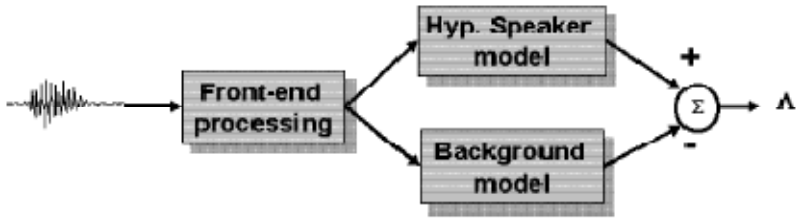


Imagen 3-7. Verificador GMM basado en verosimilitud

Recordamos que la decisión, de acuerdo con el criterio de máxima verosimilitud (Maximum Likelihood, ML) se obtiene mediante el cociente de verosimilitudes que viene dado por:

$$\frac{P(X \mid H_0)}{P(X \mid H_1)} \begin{cases} \geq \theta & \text{aceptar } \mathrm{H}_0 \\ < \theta & \text{rechazar } \mathrm{H}_0 \end{cases} \tag{3.6}$$

Aunque, como ya vimos, se suele emplear el logaritmo:

$$\Lambda(x) = \log P(X \mid H_0) - \log P(X \mid H_1) \quad \begin{cases} \geq \log\theta & \text{aceptar } H_0 \\ < \log\theta & \text{rechazar } H_0 \end{cases} \tag{3.7}$$

3.1.2.1 Cálculo de verosimilitudes

Dada una señal de voz de entrada al sistema, tras la parametrización se obtiene un conjunto de N vectores, donde N es el número de muestras de la señal de voz, y cada vector contiene D coeficientes MFCC. La función densidad de probabilidad que permite obtener la verosimilitud de haber sido generado por un determinado modelo, para cada vector, viene dada por:

$$p(x_j \mid \lambda) = \sum_{i=1}^{M} w_i \cdot p_i(x_j) \qquad 1 \leq \mathrm{j} \leq \mathrm{N} \tag{3.8}$$

La verosimilitud se calcula como la combinación lineal ponderada de la función de densidad de M Gaussianas unimodales, $p_i(x)$. Siendo x el vector de parámetros, λ el modelo que se estudia y w_i el peso de cada Gaussiana. Los coeficientes w_i son tales que cumplen $\sum_{i=1}^{M} w_i = 1$.

Como se indicaba en el apartado de entrenamiento, cada Gaussiana i viene modelada por un vector de medias μ_i (de dimensión Dx1), una matriz de covarianzas Σ_i (de dimensión D×D) y por el peso de la mezcla w_i. Siendo la expresión de la densidad de cada Gaussiana:

$$p_i(x_j) = \frac{1}{(2\pi)^{D/2} |\Sigma_i|^{1/2}} \exp\left\{ -\frac{1}{2} (x_j - \mu_i)' (\Sigma_i)^{-1} (x_j - \mu_i) \right\} \qquad 1 \leq \mathrm{j} \leq \mathrm{N} \tag{3.9}$$

La expresión anterior se simplifica si se considera sólo la diagonal de la matriz de covarianzas. Esta simplificación se realiza en base a tres razones:

- La función de densidad que se modela con la matriz de covarianzas completa de orden M puede ser modelada por la matriz diagonal si M es grande.
- Trabajar con la matriz diagonal es computacionalmente menos costoso.
- Empíricamente se ha observado que el empleo de la matriz diagonal proporciona mejores resultados.

Empleando dicha simplificación y sabiendo que σ_i es el vector de elementos de la diagonal principal de la matriz Σ_i, la función de densidad de cada Gaussiana queda:

$$p_i(x_j) = \frac{1}{(2\pi)^{D/2} \sqrt{\prod_{k=1}^{D} \sigma_{ik}^2}} \exp\left\{ -\frac{1}{2} \sum_{k=1}^{D} \frac{(x_{jk} - \mu_{ik})^2}{\sigma_{ik}} \right\} \qquad 1 \leq \mathrm{j} \leq \mathrm{N} \tag{3.10}$$

Hasta este punto se ha obtenido la verosimilitud de que una sola muestra pertenezca a un determinado modelo. Si se considera que los vectores que representan la señal de voz X = {x_1, x_2,…, x_T} son independientes, la expresión de la verosimilitud viene dada por:

$$P(x \mid \lambda) = \prod_{j=1}^{N} p(x_j \mid \lambda) \tag{3.11}$$

Calculando el logaritmo en ambos términos de la ecuación se obtiene que el logaritmo de la verosimilitud de que un fragmento haya sido producido por un determinado modelo es igual a:

$$\log P\left(x \mid \lambda\right) = \sum_{j=1}^{N} \log p\left(x_j \mid \lambda\right) \tag{3.12}$$

Aplicando la fórmula anterior para los modelos del usuario a verificar y del modelo de mundo podemos calcular su diferencia. Con esta diferencia se obtiene la medida que la aplicación compara con el umbral para tomar la decisión de aceptar al locutor o rechazarlo.

3.1.3 Fase de test

Para que la fase de verificación tenga sentido se ha tenido que hacer una estimación previa del umbral óptimo que minimice la probabilidad de error del sistema. En la fase de test es donde se realiza el cálculo de este umbral. Para ello, utilizaremos las muestras de las señales de voz de las que se conoce el locutor y haremos el cálculo de la verosimilitud. En este cálculo de verosimilitud se utiliza tanto el modelo del locutor como los modelos de los usuarios que sabemos que no realizaron las locuciones.

Entradas:	Base de datos de cada usuario Modelo de cada usuario
Salidas:	Probabilidad de error de test para cada usuario Umbral óptimo para cada usuario

Tabla 3-3. Entradas y salidas de la Fase de Test

Para calcular el valor óptimo del umbral se decide hacer un barrido con diferentes valores del mismo y comprobar las prestaciones del sistema según el umbral elegido. Después de realizado este barrido y comprobadas las prestaciones obtenidas podremos obtener el valor deseado del umbral.

Las prestaciones que hay que tener en cuenta en un sistema de verificación de locutor son, básicamente, la probabilidad de error del mismo. Para definir la Probabilidad de Error se definen otros dos tipos de probabilidades:

- Probabilidad de Falsa Aceptación.
- Probabilidad de Falso Rechazo.

3.1.3.1 Probabilidad de Falsa Aceptación

La probabilidad de Falsa Aceptación es la probabilidad de que una locución que no fue pronunciada por el usuario, cuyo modelo estamos utilizando en el proceso de verificación, sea considerada por el sistema como válida.

Para hacer el cálculo de este valor se tendrán que realizar las pruebas denominadas "*de impostor*", que consisten en calcular todas las verosimilitudes de cada locución verificándola con cada modelo de **otro** usuario que no las pronunció. El valor final de FA resulta el siguiente:

$$FA(\%) = \frac{n^{o}\ falsas\ aceptaciones}{n^{o}\ pruebas\ impostor} \times 100 \tag{3.13}$$

3.1.3.2 Probabilidad de Falso Rechazo

La probabilidad de Falso Rechazo es aquella probabilidad de que una locución pronunciada por el usuario, cuyo modelo se está utilizando en la verificación, no sea considerada por el sistema como válida.

Para realizar el cálculo de este valor se tienen que hacer las pruebas denominadas *"de no impostor",* es decir, calcular todas las verosimilitudes de cada locución verificándolas con el modelo del **locutor verdadero**. El valor final de FR resulta el siguiente:

$$FR(\%) = \frac{n^{o}\ falsos\ rechazos}{n^{o}\ pruebas\ no\ impostor} \times 100 \tag{3.14}$$

3.1.3.3 Elección del umbral de decisión

El umbral $\log\theta$ sirve para calibrar el nivel de diferencia que deben tener los resultados obtenidos, en el cálculo de verosimilitudes, para aceptar o rechazar usuarios. Como se puede observar en la imagen siguiente, los valores de la probabilidad de Falsa Aceptación y de Falso Rechazo dependen de la elección de este umbral.

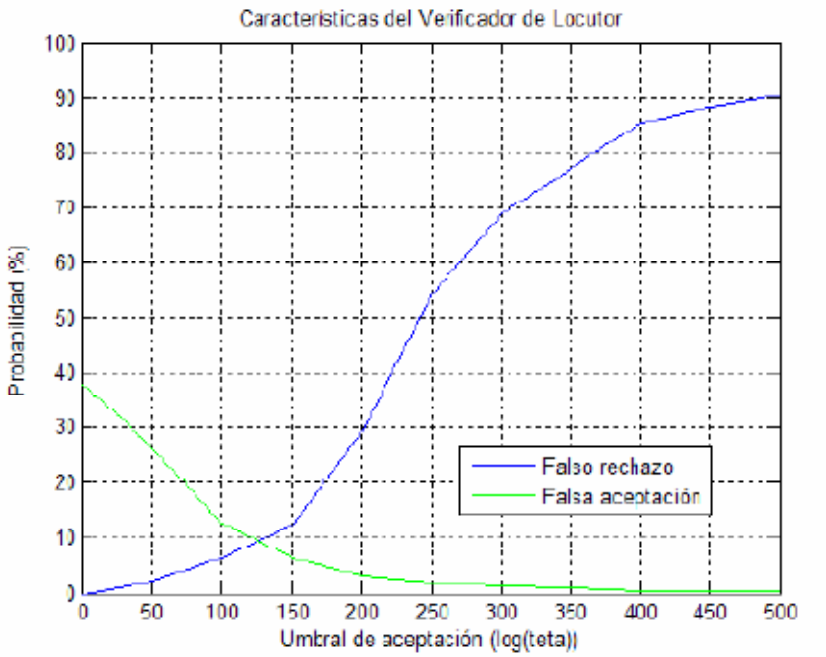


Imagen 3-8. Variación de FA y FR en función del umbral de decisión

Según aumenta el valor del umbral de comparación $\log\theta$, la probabilidad de FA disminuye (es más difícil que se acepte a un impostor) pero aumenta la probabilidad de FR (es más fácil que se rechace a un verdadero usuario). En la elección del umbral óptimo se debe calcular el valor de $\log\theta$ que minimiza la función coste de error del sistema, teniendo en cuenta que el coste de cometer cada tipo de error (FA o FR) puede ser diferente en función del ámbito de uso de la aplicación. De forma genérica, la Probabilidad de Error (PE) es la siguiente:

$$PE = c_{FA} \times FA(\theta) + c_{FR} \times FR(\theta) \tag{3.15}$$

donde c_{FA} y c_{FR} representan, respectivamente, el coste de cometer una Falsa Aceptación y el coste de cometer un Falso Rechazo.

En la realización de este proyecto se ha considerado que la Probabilidad de Error es aquella en donde se cruzan ambas curvas de FA y de FR. A este valor de PE se le denomina *EER* (Equal Error Rate).

3.1.4 Factores a tener en cuenta

A la hora de diseñar un verificador de hablante independiente de texto es necesario tener en cuenta los siguientes parámetros que van a medir la idoneidad de la implementación realizada:

- Tasa de Error. El Verificador debe proporcionar una tasa de error baja para poder considerar que el sistema es fiable. Esta probabilidad de error del sistema (EER) corresponde al porcentaje de error donde se cortan las gráficas de la probabilidad FA y de la probabilidad FR (Imagen 3-8). Los factores de los que depende el valor de estas probabilidades son:
    - Número de coeficientes MFCC.
    - Número de Gaussianas de los modelos.
    - Tamaño de palabra usada en el paso a aritmética punto fijo.
    - Tamaño de las muestras de voz.

3.2 Grabador

Dentro del proyecto global, el objetivo de este módulo es recopilar muestras de voz de un usuario, propietario de la PDA, para que el dispositivo tenga información de cómo habla el usuario y poder distinguirlo del resto del mundo.

Sin embargo, este módulo es independiente del uso del Verificador de Locutor, por lo que es posible utilizarlo como aplicación externa para los desarrolladores a la hora de hacer una base de datos para el entrenamiento del UBM.

Básicamente, la verificación de locutor es un proceso de clasificación de patrones, cuyo objetivo es clasificar la señal de entrada (onda acústica) en una secuencia de patrones previamente aprendidos y almacenados en unos modelos acústicos de usuarios. Existen muchos factores que influyen en la dificultad del proceso de verificación y reconocimiento y, por tanto, en su rendimiento, pero entre todos ellos destaca la variabilidad.

La variabilidad de la señal de voz depende tanto de factores intrínsecos al fenómeno de producción de voz, como a factores externos al mismo. Dentro de los factores intrínsecos destaca la variabilidad de los sonidos, debido fundamentalmente a los distintos acentos o formas de hablar de cada persona; y la variabilidad en la producción de los sonidos, debido fundamentalmente a las distintas velocidades de producción, coarticulación, inclusión de ruidos (apertura y cierre de labios, respiración, sonidos de duda…), condiciones acústicas (hablar en ambientes ruidosos), contexto de la conversación, estado anímico, etc.

Entre los factores externos destacan la variabilidad en la cadena de conversión y transmisión de la señal eléctrica, debido a las diferencias entre las características de los micrófonos, líneas telefónicas, etc.; y la variabilidad en el ruido captado con la señal de voz, debido a la existencia de otras fuentes sonoras en las proximidades del micrófono.

A estos factores de variabilidad acústica hay que añadir los de variabilidad lingüística relacionados con las distintas formas dialécticas de hablar un idioma, la utilización de palabras no contempladas en el vocabulario de la aplicación, la construcción de frases no permitidas por la gramática del lenguaje, la utilización de abreviaturas, los escenarios semánticos de las

palabras, etc. Todo ello hace que la verificación y reconocimiento automático del habla por parte de una máquina no sea un problema tan trivial como a primera vista pueda parecer.

Para salvar todos estos inconvenientes existe la etapa de entrenamiento, en la que el sistema aprende cómo hablan los usuarios para luego poder buscar, de entre los patrones almacenados, el que más se aproxima a la secuencia de voz de entrada. Si se quieren conseguir estos patrones de voz, será necesario recopilar muestras de la población.

Para poder subsanar los factores de variabilidad comentados, el número de registros de voz necesarios tiene que ser muy amplio y diverso, y esto no siempre es sencillo. Piense si no por un momento, en conseguir que un mínimo de 50 personas de cada sexo y de diferente índole, pasen por su despacho para grabar ciertas secuencias de voz en su PC con un determinado micrófono mientras se van etiquetando las muestras para su correcto almacenado.

3.3 Parametrizador

El módulo parametrizador se encarga de la extracción de las características propias de cada locutor que van a permitir la posterior verificación de la identidad. Para poder realizar cualquier tipo de procesado sobre las muestras de voz generadas por los locutores, es necesario realizar la extracción de la información contenida en estas muestras.

Una suposición que debe ser realizada es considerar la señal de voz estacionaria para lo que se consideran intervalos de tiempo cortos (entre 20 a 40 ms). Para trabajar en condiciones de estacionariedad se realiza un enventanado de la señal en fragmentos de 25 ms de duración. Entre fragmentos consecutivos se produce un solapamiento que permite tener una ventana de observación mayor. El análisis espectral de cada ventana temporal permite obtener los parámetros característicos de la señal de voz. Existen dos familias principales de parámetros que pueden ser extraídos: los derivados del análisis LPC (Linear Predictive Coding) [RED06], y los MFCC (Mel Frequency Cepstrum Coefficients). En la realización de este proyecto se han empleado los parámetros MFCC por implicar una menor carga computacional.

El análisis cepstral es una técnica homomórfica que permite separar la acción del tracto vocal (filtro lineal invariante en el tiempo) de la señal de excitación [ALV01] (obsérvese la Imagen 2-2). La justificación que explica que se pueda hacer este tipo de análisis de la señal de voz se resume en la siguiente figura:

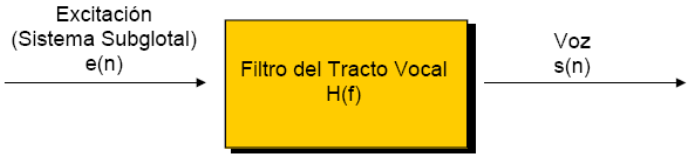


Imagen 3-9. Representación del sistema fonador como un sistema homomórfico

Para la obtención de los parámetros MFCC se ha seguido el estándar ETSI ES 201 108 [ETS03]. Los coeficientes de características obtenidos se componen de 12 coeficientes cepstrales y sus primera derivada y una medida de la energía y su primera derivada. En total, para cada fragmento de la señal de voz se obtiene un vector de 26 coeficientes MFCC. Aunque no se contempla en el estándar, se ha realizado una operación de normalización de los parámetros MFCC.

En el siguiente diagrama se muestra el esquema de las operaciones que se realizan en el análisis cepstral de la señal de voz:

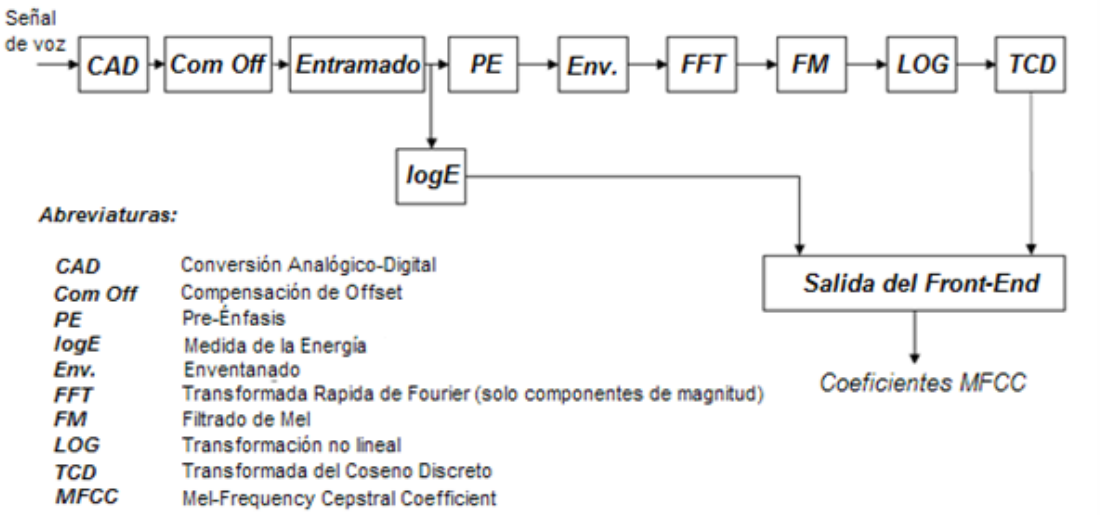


Imagen 3-10. Etapas del algoritmo de extracción de parámetros MFCC

3.3.1 Conversión analógico-digital

La señal de voz es muestreada con una frecuencia de muestreo de 8 KHz, calidad telefónica, y con un tamaño de palabra de 16 bits por muestra.

3.3.2 Compensación del offset

Se aplica un filtrado para eliminar la componente DC de la señal, proveniente del muestreado.

$$s_{of}(n) = \mathrm{s}_{\mathrm{in}}(n) - \mathrm{s}_{\mathrm{in}}(n-1) + 0.999 \times s_{of}(n-1) \tag{3.16}$$

3.3.3 Entramado

Como ya se ha indicado al inicio de este apartado, es necesario procesar la señal de voz en tramas de corta duración para poder considerar estacionalidad. Para la frecuencia de muestreo de 8 KHz, el estándar establece el tamaño de la trama en 200 muestras, lo que equivale a 25 ms, y el solape entre tramas en 80 muestras, lo que equivale a 10 ms.

3.3.4 Cálculo de la medida de la energía

El logaritmo de la energía de cada trama es uno de los coeficientes que debe ser extraído:

$$\log E = \ln\left(\sum_i s_{of}(i)^2\right) \tag{3.17}$$

donde *N* es la longitud de la trama y s_{of} es la señal de entrada con compensación del offset.

Para el cálculo, se utiliza un valor mínimo del argumento del logaritmo que asegura que el resultado es no menor de -50.

3.3.5 Pre-énfasis

Debido a las características de las cuerdas vocales, la señal de voz sufre una atenuación con la frecuencia (6 dB/octava). El filtro de pre-énfasis tiene la misión de compensar esta caída y

hacer que el espectro presente un aspecto más plano. El filtro de pre-énfasis está definido por la siguiente expresión:

$$s_{pe}(n) = s_{of}(n) - 0.97 \times s_{of}(n-1) \tag{3.18}$$

3.3.6 Enventanado

El proceso de enventanado de la señal de voz se puede considerar como la multiplicación de la misma por una señal rectangular. En el dominio frecuencial será el resultado de convolucionar el espectro de la señal de voz con el espectro de señal rectangular [OPP89]. Para paliar la aparición de componentes en alta frecuencia, debidas a las discontinuidades de la señal rectangular, se aplica una ventana de Hamming en el proceso de enventanado de la señal.

La expresión de la ventana de Hamming empleada es:

$$s_w(n) = \left[0.54 - 0.46 \times \cos\left(\frac{2\pi(n-1)}{N-1} \right) \right] \times s_{pe}(n), \qquad 1 \le n \le N \tag{3.19}$$

En la fórmula anterior, *N* representa la longitud de la trama y s_{pe} y s_w son la entrada y la salida del bloque de enventanado, respectivamente.

En las siguientes imágenes se muestra la representación de las ventanas rectangular y de Hamming, en el dominio del tiempo y de la frecuencia.

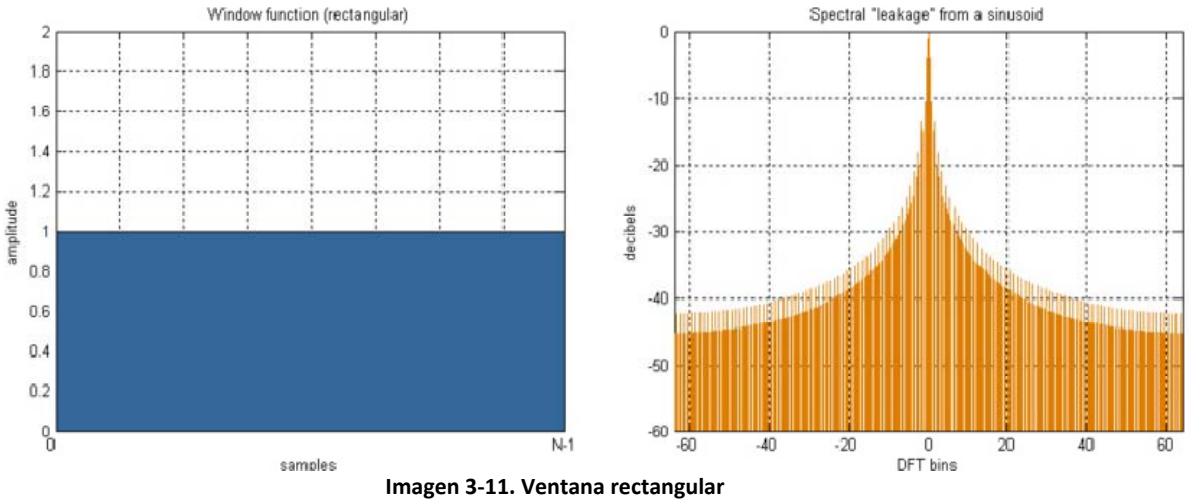


Imagen 3-11. Ventana rectangular

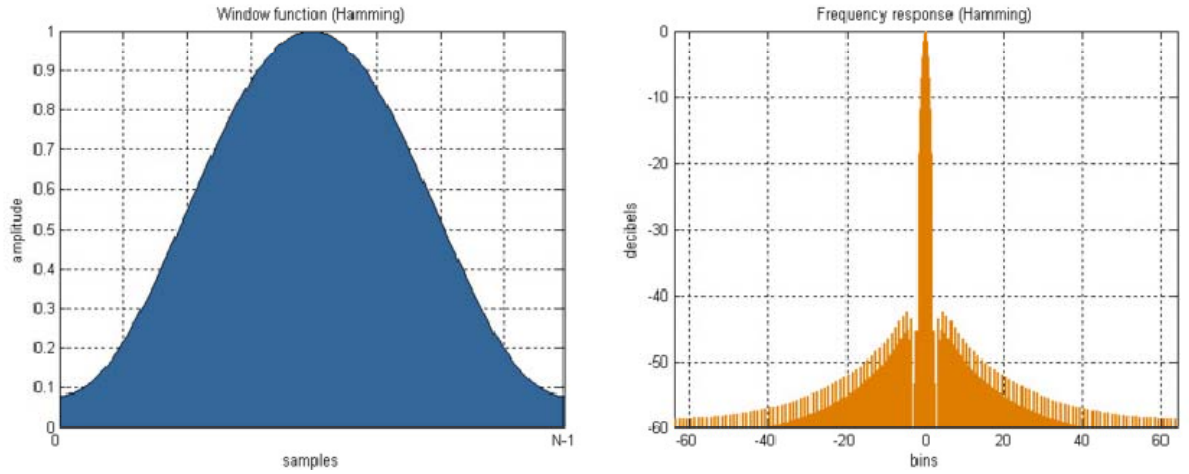


Imagen 3-12. Ventana de Hamming

Se puede observar que el espectro de la ventana de Hamming se asemeja más a una delta en el dominio de la frecuencia, por lo que la distorsión introducida en el espectro de la señal de voz es menor que en el caso de la ventana rectangular.

3.3.7 Transformada rápida de Fourier

Los segmentos fonéticos de la señal de voz son fluctuaciones de la energía a lo largo del tiempo en diferentes bandas frecuenciales. Estos parámetros pueden ser calculados computando el módulo de la Short Time Fourier Transform (STFT). El resultado obtenido es una estimación de la densidad espectral de potencia de la señal de voz, habiendo asumido que la misma es localmente estacionaria.

Según el estándar, y para la frecuencia de muestreo empleada (8 KHz) cada trama se rellena con muestras cero hasta conseguir una longitud de 256 muestras. A partir de esta trama extendida se calcula su transformada rápida de Fourier (FFT). La expresión empleada para calcular la FFT se muestra a continuación:

$$bin_k = \left| \sum_{n=0}^{FFTL-1} s_w(n) \times e^{\left(-jnk\frac{2\pi}{FFTL} \right)} \right| \qquad k = 0,\ldots,FFTL-1 \tag{3.20}$$

En ella, s_w es la trama de entrada, FFTL es la longitud del bloque de FFT y bin_k es el valor absoluto del vector complejo de salida.

NOTA: Debido a la simetría, solo se utilizan $bin_{0\ldots FFTL/2}$.

3.3.8 Filtrado Mel

Los componentes de frecuencia baja del espectro son ignorados. La banda de frecuencia útil está entre 64 Hz y la mitad de la frecuencia de muestreo actual. Este grupo está dividido en 23 canales equidistantes en escala de frecuencia mel. Cada canal tiene forma triangular. Los canales consecutivos se superponen a la mitad.

La opción de la frecuencia inicial del banco de filtros, f_{start} = 64 Hz, aproximadamente equivale al caso donde toda la banda de frecuencia está dividida en 24 canales y el primer canal es desechado usando cualquiera de las tres frecuencias de muestreo permitidas en el estándar.

Las frecuencias centrales de los filtros en términos de índices de FFT ($cbin_i$ para el canal *i-ésimo*) se calculan como sigue:

$$Mel\{x\} = 2595 \times \log_{10}\left(1 + \frac{x}{700}\right) \tag{3.21}$$

$$f_{c_i} = Mel^{-1}\left\{ Mel\{f_{start}\} + \frac{Mel\{f_s/2\} - Mel\{f_{start}\}}{23+1} \cdot i \right\} \qquad i = 1,\ldots,23 \tag{3.22}$$

$$cbin_i = round\left\{ \frac{f_{c_i}}{f_s} FFTL \right\} \tag{3.23}$$

donde *round{x}* se utiliza para redondear hacia el número entero más cercano.

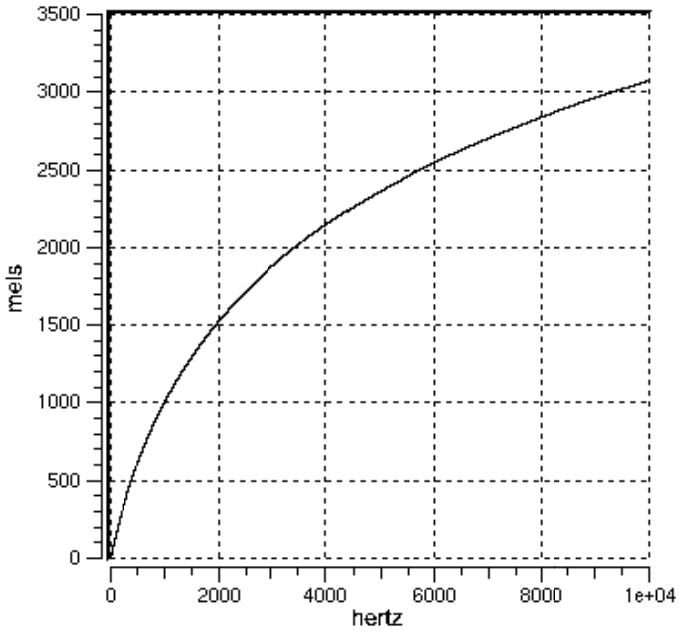


Imagen 3-13. Escala de Mel

La salida de cada filtro es la suma ponderada de los valores del espectro de magnitud FFT (bin_i) en cada grupo. El enventanado triangular, medio solapado se usa como sigue:

$$fban_k = \sum_{i=cbin_{k-1}}^{cbin_k} \frac{i - cbin_{k-1} + 1}{cbin_k - cbin_{k-1} + 1} \cdot bin_i + \sum_{i=cbin_k+1}^{cbin_k} \left(1 - \frac{i - cbin_k}{cbin_{k+1} - cbin_k + 1}\right) \cdot bin_i \quad (3.24)$$

donde k = 1,…,23, $cbin_0$ y $cbin_{24}$ denotan los índices bin de la FFT correspondientes a la frecuencia inicial y a la mitad de la frecuencia de muestreo, respectivamente,

$$cbin_0 = round\left\{\frac{f_{start}}{f_s} FFTL\right\} \quad (3.25)$$

$$cbin_{24} = round\left\{\frac{f_s/2}{f_s} FFTL\right\} = FFTL/2 \quad (3.26)$$

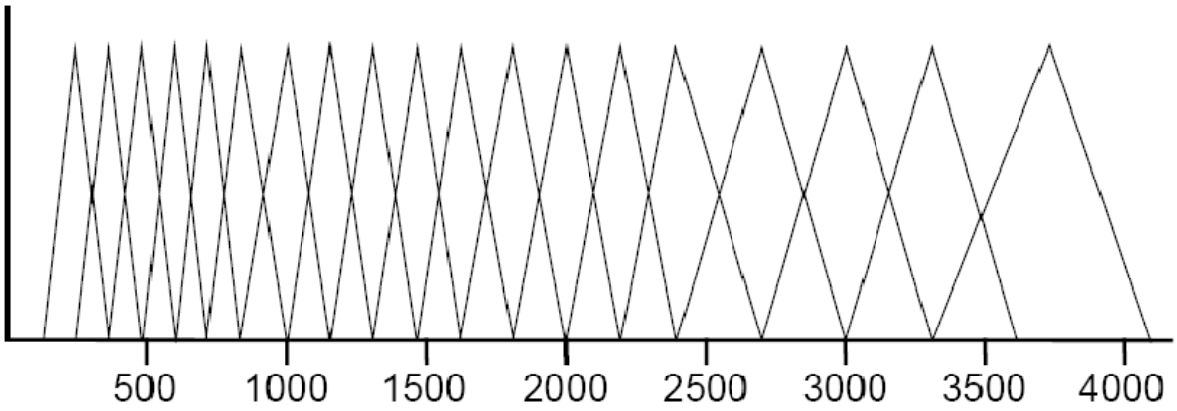


Imagen 3-14. Banco de filtros de Mel

3.3.9 Transformación no lineal

La salida del filtrado del banco de filtros en escala mel está sujeta a una función logarítmica (logaritmo neperiano).

$$f_i = \ln\left(fbank_i \right), \qquad i = 1,...,23 \tag{3.27}$$

El mismo suelo es aplicado que en caso del cálculo de energía, es decir, las salidas del banco de filtros no pueden ser más pequeñas que -50.

3.3.10 Coeficientes Cepstrales

A la salida del bloque de Transformación no lineal se calculan 13 coeficientes cepstrales usando la transformada de coseno discreta (DCT).

$$C_i = \sum_{j=1}^{23} f_j \times \cos\left(\frac{\pi \times i}{23}\left(j - 0.5 \right) \right), \qquad 0 \le i \le 12 \tag{3.28}$$

3.3.11 Salida del Front-End

El vector final consiste en 14 coeficientes: el coeficiente del logaritmo de la energía (punto 3.3.4) y los 13 coeficientes cepstrales (punto 3.3.10).

El coeficiente C_0 es redundante cuando el coeficiente del logaritmo de la energía ha sido calculado por lo que se obtienen únicamente 13 coeficientes cepstrales.

Finalmente se calcula la primera derivada de los coeficientes de salida obtenidos, para tener en cuenta la dependencia de cada trama con la anterior, como medida de la correlación existente entre ellas. A estos parámetros se les denomina como coeficientes delta. Una vez obtenidos estos coeficientes se logra tener un vector de veintiséis parámetros.

El método para su extracción se presenta en la ampliación del estándar seguido hasta el momento, el estándar ETSI ES 202 050 [ETS07]. Para su obtención se emplea la expresión:

$$\begin{aligned} d(i,t) = & -c(i,t-4) - 0.75 \times c(i,t-3) - 0.5 \times c(i,t-2) - 0.25 \times c(i,t-1) \\ & + 0.25 \times c(i,t+1) + 0.5 \times c(i,t+2) + 0.75 \times c(i,t+3) + c(i,t+4) \\ & 1 \le i \le 12 \end{aligned} \tag{3.29}$$

3.4 Adaptador

El objetivo de la fase de entrenamiento es obtener los modelos necesarios para realizar la verificación de locutor. Se debe generar un modelo independiente del locutor, el modelo de mundo UBM, y un modelo dependiente del locutor para cada usuario, el modelo de usuario. De esta forma, con el modelo de mundo y el de usuario, se podrá realizar la comparación en la fase de test.

La adaptación es un proceso decisivo que se encuentra situado dentro de la fase de entrenamiento anteriormente comentada. Se tratará de resolver el problema de la adaptación con la mayor eficiencia posible, de modo que se pueda obtener un buen modelo de usuario a partir del modelo de mundo sin un gran gasto computacional. El proceso se debe realizar en la PDA, actualizando el modelo independiente del locutor con un conjunto de muestras de voz

del locutor correspondiente, convirtiendo el modelo de mundo en un modelo de usuario, ahora dependiente del locutor.

Imagen 3-15. Proceso de adaptación del verificador

3.4.1 Modelo de Mundo

En el sistema de verificación GMM-UBM, se utiliza un único modelo independiente del interlocutor que se denotará por UBM (Modelo Universal de Mundo o Universal Background Model). Se entrena para representar a todos los interlocutores de la población, de forma que los atributos de este modelo sean independientes del individuo. Hay que encontrar entonces muestras de voz que reflejen las alternativas esperadas durante el proceso de reconocimiento, tanto en tipo y en calidad de voz, como en la composición de interlocutores. No es necesario que sea un conjunto de muestras masivo, es más conveniente que sea pequeño y representativo.

A la hora de entrenar el UBM, se debe tener en cuenta que no existen medidas objetivas que determinen el número óptimo de hablantes que se debe emplear ni la duración de las locuciones. Sí que se tienen que considerar las características de la población que representará al modelo. Por ejemplo, se puede desear un UBM para mujeres y otro para hombres, o uno para niños y otro para adultos. En el presente, se va a modelar un único modelo (ver Imagen 3-3) que represente a todos los hablantes considerados, por ello se debe prestar atención en que los datos empleados para el entrenamiento estén balanceados para las diferentes subpoblaciones.

3.4.2 Modelo de Usuario: Adaptación

La adaptación es un problema crucial en el reconocimiento y la verificación actual. La adaptación del locutor se utiliza para obtener un buen modelo de usuario a partir del modelo de mundo, actualizando el modelo independiente del locutor con un conjunto de muestras de voz del locutor correspondiente. De este modo se convierte el UBM en un modelo dependiente del locutor.

Hay diferentes aproximaciones para llevar a cabo la adaptación de usuario, las más utilizadas recientemente son la cuantificación vectorial (VQ, [SOO85]), la regresión lineal de máxima probabilidad (MLLR, [DOH00]) y el máximo a posteriori (MAP, [GRO70]). Reynolds propuso además una variante de adaptación Bayesiana, la adaptación basada en el modelo universal, UBM, muy utilizado en la actualidad [REY00].

Es evidente que los sistemas que obtienen el modelo de usuario únicamente a partir de grabaciones de ese locutor, alcanzan mejores resultados, pues el esquema se ajusta mucho más. No obstante, tienen el claro inconveniente de que se necesita una base de datos del usuario más extensa para conseguir un modelo satisfactorio. Aplicando técnicas de adaptación

en vez de entrenamiento de modelos específicos, se pueden obtener modelos de usuario que proporcionen resultados satisfactorios con tan sólo unas pocas muestras de voz del locutor.

En el sistema GMM-UBM que se está viendo, se deriva entonces el modelo del interlocutor adaptando los parámetros del modelo de mundo UBM. Para ello, se emplea una estimación MAP (*Máximum A Posteriori* o aprendizaje Bayesiano) utilizando como entrada tramos de voz del interlocutor en cuestión. La idea es obtener el modelo de usuario actualizando los parámetros del UBM (que en su momento ya habían sido bien entrenados para su población) por medio de la adaptación. Esto provoca un mayor acoplamiento entre el modelo UBM y los modelos de los locutores, con lo que se consiguen mejores resultados que los modelos desacoplados y permite emplear técnicas para agilizar el cálculo de la verosimilitud en la fase de test.

En el proceso de adaptación del modelo de mundo, no se suelen adaptar todas las Gaussianas del UBM, por tanto, es posible crear un modelo simplificado que solamente guarde los valores de aquellas Gaussianas que han sufrido modificaciones.

3.4.3 Algoritmo de adaptación GMM-UBM

El algoritmo de adaptación cuenta con dos pasos principales, al igual que el algoritmo EM presentado en el apartado 2.4.3.2.3. El primer paso es el mismo, y en él se calculan las estimaciones de los estadísticos suficientes para cada una de las mezclas del UBM. Sin embargo, en el segundo paso del algoritmo de adaptación, se combinan estos nuevos estadísticos con los anteriores (de los parámetros de la mezcla del UBM) ponderados por los coeficientes de mezcla, dependientes de los datos. Estos coeficientes se diseñan de tal forma que aquellas mezclas con más cantidad de muestras del interlocutor dependen más de los nuevos estadísticos suficientes para la estimación final de parámetros que los que tienen menor número de datos, que dependerán en mayor medida de los viejos estadísticos para la estimación final de los parámetros.

El proceso es el siguiente:

1. **Paso E.** Dado un UBM y un conjunto de vectores de entrenamiento del interlocutor propuesto X = {x_1, ..., x_T }, lo primero es ubicar los vectores en el espacio de las mezclas UBM, esto es, para la componente *i* en el UBM, se calcula:

$$\Pr\left(i \mid x_i\right) = \frac{w_i p_i\left(x_i\right)}{\sum_{j=1}^{M} w_j p_j\left(x_i\right)} \tag{3.30}$$

   donde se recuerda que los w_i son las ponderaciones de las Gaussianas $p_i(x_i)$, definidas en la ecuación (3.9) y M el número de Gaussianas en la combinación.

   Con $Pr(i|x_i)$ se procede a calcular los estadísticos suficientes para los parámetros del peso, media y varianza para cada mezcla. Los dos últimos son vectores con el mismo tamaño que los vectores de entrenamiento x_t, 1 × D:

$$n_i = \sum_{t=1}^{T} \Pr\left(i \mid x_t\right) \tag{3.31}$$

$$E_i\left(x\right) = \frac{1}{n_i} \sum_{t=1}^{T} \Pr\left(i \mid x_t\right) \cdot x_t \tag{3.32}$$

$$E_i\left(x^2\right)=\frac{1}{n_i}\sum_{t=1}^{T}\Pr\left(i\mid x_t\right)\cdot x_t^2 \qquad (3.33).$$

2. **Adaptación.** Se utilizan los nuevos estadísticos de los datos de entrenamiento, para actualizar los de la mezcla *i* del UBM y obtener así los parámetros adaptados de la mezcla *i*:

$$\widehat{w_i}=\left[\alpha_i^w n_i/T+\left(1-\alpha_i^w\right)w_i\right]\gamma \qquad (3.34)$$

$$\widehat{\mu_i}=\alpha_i^m E_i\left(x\right)+\left(1-\alpha_i^m\right)\mu_i \qquad (3.35)$$

$$\widehat{\sigma_i}=\alpha_i^v E_i\left(x^2\right)+\left(1-\alpha_i^v\right)\left(\sigma_i^2+\mu_i^2\right)-\widehat{\mu}_i^2 \qquad (3.36)$$

Los coeficientes de adaptación que controlan el balance entre los nuevos y los viejos estadísticos son $\left\{\alpha_i^w,\alpha_i^m,\alpha_i^v\right\}$ para los pesos, medias y varianzas, respectivamente.

El factor de escala γ se calcula sobre todos los pesos de mezclas adaptados para asegurar que sumen la unidad, i.e. $\sum_{i=1}^{M}\widehat{w_i}=1$.

Para cada Gaussiana y para cada parámetro, se utiliza un coeficiente de adaptación dependiente de los datos de entrenamiento, α_i^ρ con $\rho\in\left\{w,m,v\right\}$, que se calcula como $\alpha_i^\rho=\frac{n_i}{n_i+r^\rho}$, siendo r^ρ un factor de relevancia para el parámetro ρ. El uso de estos coeficientes permite una adaptación de los parámetros dependiente de la mezcla, balanceando la ponderación hacia los nuevos parámetros, o hacia los viejos.

CAPÍTULO IV

Implementación en PDA

El Verificador de Locutor diseñado en el desarrollo de este proyecto ha sido implementado para su portado a un dispositivo móvil, en este caso se trata de una PDA perteneciente al Grupo de Procesado Multimedia del Departamento de Teoría de Señal y Comunicaciones de la Universidad Carlos III de Madrid.

De esta forma, un usuario que compre el Verificador puede disponer en el dispositivo de un modelo personalizado para que autentique su voz. Para ello, no tendrá más que grabar una serie de frases con el módulo Grabador, que servirán de entrada para este programa de adaptación. En el paquete de la aplicación se encontrará incluido el modelo de mundo, por lo que únicamente se han de extraer los parámetros de las secuencias de voz *.wav* grabadas gracias al Parametrizador implementado y adaptar el modelo UBM con la información contenida en ellas utilizando para ello el módulo Adaptador.

El Modelo General de Mundo UBM será proporcionado al usuario final gracias a un proceso de entrenamiento realizado dentro del Laboratorio en el que se han utilizado los datos de una Base de Datos interna del Departamento en la que un total de veintiocho locutores han grabado su voz en 3 sesiones de 50 palabras (las mismas para todos los locutores) cada una. Para este proceso se ha utilizado la aplicación HTK.

4.1 Proceso de implementación

El proceso de diseño del Verificador desarrollado en este proyecto ha tenido varias fases diferentes hasta alcanzar la última versión en la PDA. Los primeros pasos se centraban en la implementación de un Verificador en Punto Flotante para un ordenador de sobremesa.

Una vez evaluado el comportamiento de esta versión, el proceso de implementación pasó a centrarse en el traslado de la aplicación de una aritmética en punto flotante a otra en Punto Fijo. Este paso se dio pensando que el futuro de la aplicación iba a ser su portado a una PDA cuyas prestaciones son sensiblemente inferiores a las de un PC.

El último paso en todo el proceso fue adaptar todas las aplicaciones implementadas de manera que funcionaran en la PDA. Para ello hubo que realizar diversas modificaciones en el código de cada aplicación debido a las diferencias existentes entre una PDA y un PC.

4.1.1 Plataforma hardware

4.1.1.1 Origen

La aparición de la primera PDA corrió a cargo de la compañía Apple Computers cuando presentó el primer prototipo en una feria tecnológica estadounidense a comienzos del año 1992. Fue una propuesta que no solo no llamó la atención del público, sino que además supuso uno de los fracasos más notables de su historia, porque, a pesar de que ya incorporaba la mayoría de características que tiene cualquier PDA moderna (conexión a redes, reconocimiento de escritura, etc.), éstas estaban poco desarrolladas todavía y presentaban fallos importantes.

A mediados de los 90, Apple Computers quiso resarcirse de tal varapalo y vendió la patente de su fracasado invento a Palm, empresa fundada por Ed Colligan. Así, a principios de 1996 la compañía puso a la venta su primera generación de PDA, las Palm Pilot, muy primitivas, con pantalla monocroma, procesador Motorola, y sistema operativo propio (Palm OS, desarrollado por la división Palm Source). Gracias a su tesón y esfuerzo, Palm devolvió el prestigio perdido a las PDA, alzándose ella misma como una de las empresas punteras del sector.

Microsoft no pudo resistir ante la tentación de fabricar y comercializar un dispositivo que comenzaba a tener futuro, y así, en 1998, la compañía de Bill Gates irrumpió en el mercado de las PDA con su modelo Palm PC, que introducía una variante adaptada del sistema operativo Windows. Poco después pasaría a llamarse Pocket PC, como consecuencia lógica del proceso judicial abierto contra Microsoft por la empresa Palm.

La lucha entre los dos gigantes estaba servida hasta que a mediados de 2005, año en el cual se anuncia un acuerdo de colaboración entre Microsoft (encargada del sistema operativo) y Palm (fabricante del hardware) tras la venta por parte de esta última de su división Palm Source a la empresa japonesa Access LTD.

A día de hoy se pueden encontrar Palm con el sistema operativo de Microsoft, sobre todo en los Smartphone, dispositivos que integran PDA y teléfono, hacia los que está evolucionando Palm.

4.1.1.2 Pocket PC

De acuerdo a la reseña publicitaria de Microsoft, se puede definir al Pocket PC como un dispositivo de mano (*Palm Device*) capaz de grabar mensajes de voz, escribir, enviar y recibir e-mails, contactos y citas, mostrar archivos multimedia, juegos, aplicaciones, intercambiar mensajes con MSN Messenger2, navegar por la web y mucho más.

Pero mucho más allá de cualquier reclamo publicitario, lo cierto es que el Pocket PC se constituye en sí mismo como un estándar (en parte porque Microsoft no es el responsable de la fabricación directa de los dispositivos), por lo que, para poder decir que una cierta PDA es realmente un Pocket PC, han de cumplirse una serie de requisitos y especificaciones, tanto a nivel hardware como software:

- Su sistema operativo es Microsoft Windows CE o Windows Mobile.
- Basa su arquitectura en procesadores StrongARM (una variante de los ARM tradicionales) o más recientemente en los compatibles con ARM, Intel XScale.
- Tiene aplicaciones en la ROM, esto es, en la memoria de sólo lectura interna.
- Dispone de pantalla táctil y de un puntero para interaccionar con ella.
- Dispone de pulsadores hardware de acceso directo a las aplicaciones.

Este tipo de PDA dispone de mayores posibilidades multimedia comparando con Palm, pero con la desventaja de que su precio y su consumo energético es mucho mayor.

Windows CE es el sistema operativo, actualmente va por la versión 6.0 y existen lo que se llama conjuntos de adaptación (del inglés *adaptation kits*) a Smartphone, Pocket PC, Media Player, etc. Estos *adaptation kits* sólo están disponibles mediante una licencia OEM[10] con Microsoft.

Aclarar que Windows Mobile es una implementación particular de Windows CE. Se puede instalar en dispositivos que cumplan con las especificaciones «Pocket PC» de Microsoft.

4.1.1.3 HP iPack Pocket PC serie h4100

La PDA facilitada para realizar las pruebas del programa es una iPAQ de HP, concretamente la serie h4100. Las especificaciones más importantes son:

- Procesador: 400 MHz XScale basado en tecnología de procesadores Intel.
- Memoria: 64MB de RAM (55MB de memoria principal).
- Sistema Operativo: Pocket PC 2003 Premium con el Microsoft Outlook 2002.
- Tamaño: 11,4 x 7,06 x 1,35 cm.
- Peso: 132 g.
- Conectitividad: LAN (802.11b), Bluetooth v1.1 e infrarojos.
- Expansión: Ranura SD *(Secure Digital)* compatible con SDIO *(Secure Digital Input Output)* y MMC *(Multi Media Card)*.
- Pantalla: TFT[11] de 3'5 pulgadas, 65.536 colores.
- Multimedia: Micrófono y altavoz integrados, conector de audio estéreo.

4.1.2 Programación en PDA

El sistema operativo de la PDA soporta el familiar *API*[12] de Win32, y este es el factor más significativo del sistema operativo. De este modo, se asegura que los programadores con conocimientos previos de programación en Windows se adapten rápido a este nuevo entorno, ya que les va a resultar muy familiar.

Debido a que Windows CE es mucho más compacto y tiene una estructura de hardware diferente, no toda la API de Win32 está soportada y aunque sí la parte más importante,

[10] Software *OEM* (siglas de Original Equipment Manufacturer) es un tipo de licencia para software preinstalado en equipos nuevos. Este software está ligado a la vida útil del ordenador.

[11] *TFT*: Transistor de Película Fina o Thin Film Transistor.

[12] *API* (del inglés Application Programming Interface o Interfaz de Programación de Aplicación) es el conjunto de funciones y métodos que ofrece cierta librería para ser utilizado por otro software como una capa de abstracción.

algunas están reducidas. Estas diferencias se deben a la potencia, memoria y forma de mostrar los datos de los dispositivos para los que está diseñado este sistema. También hay algunas funciones añadidas a la API y que sólo existen en Windows CE y son completamente nuevas. Los cambios más significativos con la API están relacionados con el GDI[13].

Hay dos modos de desarrollar aplicaciones para Windows CE, la primera es programar todo desde cero de forma independiente, utilizando las capacidades de la API, y la segunda usando las Microsoft Foundation Classes (MFC). En este proyecto se ha optado por la segunda opción, pues reutiliza los componentes ya programados que vienen con el compilador. MFC es una librería de clases, un conjunto de clases C++ que han sido escritas para reducir la cantidad de trabajo que tomaría realizar ciertas cosas utilizando la API. La MFC además introduce dentro de la aplicación un esqueleto orientado a objetos, del cual se puede tomar ventaja.

4.1.2.1 Memoria

La memoria juega un papel vital en el desarrollo de aplicaciones. Normalmente los programadores tienden a pasar por alto este aspecto, debido a la gran capacidad de memoria que tienen los ordenadores de hoy en día, pero los dispositivos que usan Windows CE van a tener, por lo general, una cantidad de memoria mucho menor. De esta misma manera, tampoco van a tener unas unidades de disco duro ni otros dispositivos de almacenamiento masivo más allá de las tarjetas de memoria extraíbles. Por todo ello, se ha tenido muy presente este aspecto a la hora de desarrollar la aplicación para la PDA.

Generalmente, los dispositivos con Windows CE, como es el caso del utilizado para este proyecto, van a tener una sección de memoria *RAM*[14] y otra *ROM*[15].

- Memoria ROM contiene toda la información decidida por el fabricante, la cual incluye el sistema operativo actual y otras aplicaciones preinstaladas.
- Memoria RAM tiene dos partes bien diferenciadas:
    - Memoria para programas (*Program Memory*) Se emplea para la ejecución de aplicaciones. Se puede comparar con la memoria RAM de los ordenadores normales, donde se cargan las aplicaciones ejecutadas.

    - Memoria de almacenamiento (*Storage Memory* u *Objetc Store*). La memoria de almacenamiento se usa principalmente para simular un disco duro.
      La memoria de almacenamiento o almacén de objetos es el sitio usado para mantener el sistema de archivos, el registro y las bases de datos.

4.1.2.2 Energía

Las aplicaciones para Windows CE deben de ser desarrolladas de manera que impliquen un bajo coste computacional, puesto que los dispositivos para los que estará dirigida la aplicación funcionarán mediante baterías.

Las operaciones de una aplicación que impliquen un alto gasto de energía deben ser reducidas al mínimo. Posibles ejemplos de esto son el uso abusivo de gráficos, cálculos complejos que supongan muchos bucles, etc.

En resumen, la eliminación de un uso innecesario de ciclos de CPU ahorra energía.

[13] *GDI*: Graphics Device Interface, motor gráfico de los sistemas Windows de Microsoft.
[14] *RAM* es el acrónimo del inglés Random Access Memory o Memoria de Acceso Aleatorio.
[15] *ROM* es el acrónimo del inglés Read-Only Memory o Memoria de Sólo Lectura.

4.1.2.3 Interfaz de usuario

La interfaz de usuario de Windows CE difiere del tradicional UI (*User Interface*) encontrado en las otras versiones de Windows. Una de las diferencias más obvias es el reemplazo de las barras de menú y barras de herramientas por una única barra que contiene a las dos, para dejar más espacio a la ventana principal de los programas. La inclusión de esta nueva característica ha dado lugar a una nueva API que no está en el resto de Sistemas Operativos de Windows. En otras palabras, es una API exclusiva de Windows CE.

Otro cambio notable es el puntero con el que el usuario lo maneja prácticamente todo, que reemplaza al ratón de los PC tradicionales. Este puntero se usa sobre la pantalla táctil y el principal problema que supone a la hora de programar es que sólo puede representar a un botón, por tanto desaparece la funcionalidad del botón derecho del ratón.

Un usuario de PC tradicional que se ponga a manejar una PDA encontrará muy pobre el explorador de archivos disponible en el dispositivo de mano. Esto es algo que supuso un esfuerzo extra en la aplicación para escoger una carpeta o un fichero pues, por seguridad, Microsoft no deja moverse libremente por el árbol de directorios completo del aparato.

4.1.3 Software de desarrollo

Existen varias opciones para desarrollar aplicaciones sobre estos dispositivos. Casi todas disponen de un entorno de desarrollo que se instala en un PC, aunque existe alguna que permite realizar todo el trabajo desde el propio dispositivo (no para este caso). Por lo general, los entornos para PC contienen un emulador que, en teoría, debe permitir realizar las pruebas sin necesidad de disponer de un PDA. Sin embargo, es muy probable que tarde o temprano se compruebe que el emulador no es apto para probar determinadas características del proyecto. En este caso en particular falló la grabación de voz y aunque no era posible emular al completo la aplicación, sí sirvió para la primera toma de contacto con el entorno.

4.1.3.1 EMbedded Visual C++

De todas las herramientas disponibles se utilizará entonces la que ofrece el fabricante del sistema operativo de la plataforma Pocket PC 2003 SE: eMbedded Visual C++ 4.0 de Microsoft.

Éste soporta el desarrollo de aplicaciones para Windows CE .NET 4.2 (por ello, Windows Mobile 2003, y Windows Mobile 2003 Second Edition si se instala el *Service Pack 3*) y Windows CE 5.0.

EMbedded Visual C++ 4.0 (eVC4 desde ahora) es la herramienta adecuada para escribir aplicaciones en código nativo para estas plataformas. Entre las ventajas propias de eVC4, se pueden listar:

- Depuración «en tiempo real» *(Just-In-Time debugging)* para el diagnóstico de excepciones no controladas.
- Manejo estructurado de excepciones de C++.
- Capacidad de unir el depurador a un proceso para realizar una depuración más eficiente y productiva.
- Una mejor integración con el nuevo emulador.
- Es gratuita, por lo que se reduce la inversión necesaria.
- Aunque no hay prácticamente libros editados sobre este entorno para iniciarse, hay sitios en Internet y foros para consulta de dudas. Además, dispone de una ayuda muy completa en inglés y de un sitio oficial para desarrolladores, en varios idiomas, [MSD].

- Es una de las más versátiles, ya que permite crear aplicaciones para varias versiones de las plataformas Pocket PC.

El lenguaje de programación C es el que ha servido de iniciación para aprender eVC++, pues se disponía de conocimientos previos en este lenguaje. Debido a que no se encontró un manual completo de eVC++, se optó por seguir una serie de pasos evolutivos hasta alcanzar este lenguaje: se comenzó con la base, ANSI[16] C (ver [KER78]), para después pasar a C++[17] (ver [BUS04a], [BUS04b], [CEB07], [GRA92], [STR97], [POZ] y [JON04], Visual C++ (ver [SPH] y [CHA98]) y finalmente eVC++.

Después de dos meses de aprendizaje ya se dispone de preparación suficiente como para entrar a programar en eVC++ 4.0. Se comenzó realizando unos 30 programas cortos orientados a manejar el entorno de desarrollo y el emulador, algunos de ellos fruto de seguir alguno de los tutoriales de Visual C++, otros eran simples pruebas. De ellos, los más, serían útiles e integrables en el futuro programa final.

4.2 Aritmética entera

Antes de comenzar a diseñar el programa para una plataforma móvil como pueda ser una PDA se comenzó por diseñar el sistema Verificador para PC. De esta manera, el programa se podía escribir en su primera versión en punto flotante. Pero los dispositivos portátiles, como los terminales móviles, suelen presentar una CPU con una capacidad de procesamiento muy inferior a la que se puede presentar en cualquier PC de gama media.

Por esta razón, en el desarrollo para dispositivos portátiles se emplea aritmética punto fijo, ya que los algoritmos en punto fijo representan menor coste computacional y son más rápidos que sus homólogos en punto flotante. Tienen los inconvenientes de disminuir el rango dinámico con el que se puede trabajar, añadir ruido de cuantificación, aumentar la probabilidad de desbordamientos y además supone un esfuerzo extra del programador [RED06].

Para la conversión a punto fijo de los modelos y de los ficheros de características se emplearon aplicaciones C++ desarrolladas para tal efecto por el Grupo de Procesado Multimedia del Departamento de Teoria de la Señal y Comunicaciones.

4.2.1 Descripción

El uso de la aritmética punto fijo permite trabajar con números enteros como si estos tuvieran una parte decimal. El fundamento teórico sobre el que se sustenta esta aritmética es considerar que en algún punto se define una coma que separa parte entera y parte decimal.

En la Imagen 4-1 se muestra la representación binaria de un entero de 16 bits (short). Se tiene un bit de signo y 15 bits de valor que permiten representar el rango de números enteros de 32.767 a -32.768.

[16] *ANSI* (American National Standards Institute) es el Instituto Nacional Estadounidense de Estándares. Es una organización sin ánimo de lucro que supervisa el desarrollo de estándares para productos, servicios, procesos y sistemas en los Estados Unidos. ANSI es miembro de la Organización Internacional para la Estandarización (ISO) y de la Comisión Electrotécnica Internacional (IEC).

[17] *C++* es una extensión a C para soportar la programación orientada a objetos. También se le denomina Cpp del inglés C plus plus.

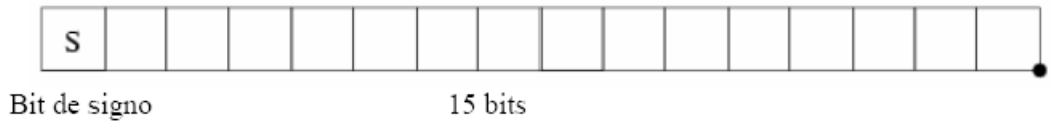


Imagen 4-1. Representación binaria de un short

Si se supone que existe una coma en algún punto, entre los 15 bits de valor, quedan definidos dos grupos de bits. Estos grupos de bits representan a la parte entera y a la parte decimal.

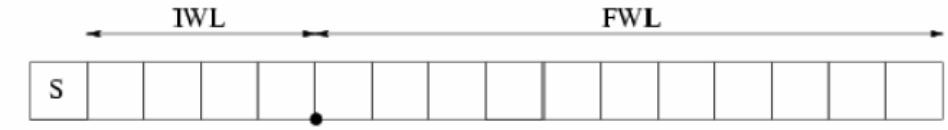


Imagen 4-2. Entero de 16 bits empleado para representar un número decimal

En la Imagen 4-2 se ha definido una coma entre los bits de valor decimoprimero y duodécimo. De esta forma se tienen 4 bits para representar la parte entera del número y 11 bits para representar la parte decimal. Al tamaño de la palabra usado para representar la parte entera se le denomina IWL (Integer Word Length) y al usado para representar la parte decimal FWL (Fractional Word Length).

El rango de valores que puede ser representado con estos 16 bits en aritmética punto fijo va de 15,999512 a -16,0 con una precisión de 2^{-11}. El rango de valores representables (R) y el valor del paso de cuantificación (Q) dependen del tamaño de la parte decimal y de la parte fraccional según las fórmulas:

$$Q = 2^{-FWL} = 2^{WL-IWL-1} \tag{4.1}$$

$$-2^{IWL} \le R \le 2^{IWL} - 2^{-FWL} \tag{4.2}$$

Cuando se trabaja con aritmética punto fijo se habla de resolución Qn, donde *n* es el número de bits de la parte decimal. El caso del ejemplo de la Imagen 4-2 se trataría de aritmética Q11. Los números enteros se corresponden con la aritmética de punto fijo de resolución Q0.

Para convertir números de punto fijo a punto flotante y viceversa se emplean las siguientes expresiones:

$$Float2Int\left(X, Q_n\right) = Round\left\lfloor X \cdot \left(2^{Q_n - 1}\right)\right\rfloor \tag{4.3}$$

$$Int2Float\left(X, Q_n\right) = \left(float\right) X \cdot 2^{-Q_n} \tag{4.4}$$

A continuación se muestran algunas ideas básicas a la hora de trabajar en aritmética punto fijo:

- *Cambio de la resolución*

Un desplazamiento a izquierda o derecha de *n* bits se corresponde con un incremento o decremento, respectivamente, del tamaño de la parte decimal FWL. Esto es lo mismo que incrementar o decrementar Q en *n* unidades.

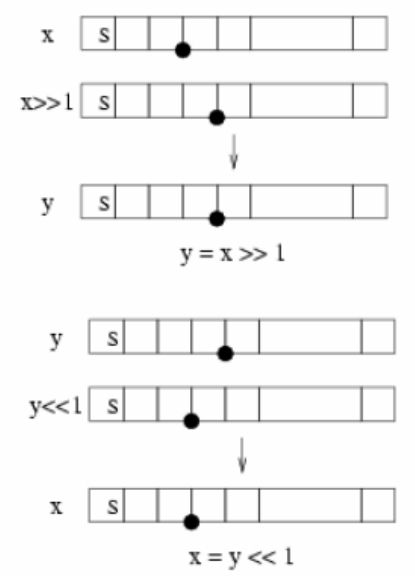


Imagen 4-3. Cambio de resolución en punto fijo

- *Realización de sumas y restas*

Para realizar sumas y restas es necesario que ambas variables se encuentren expresadas en la misma resolución Q. El resultado es una variable en la misma resolución.

En la realización de estas operaciones puede producirse un desbordamiento si el resultado no puede ser representado con la longitud de IWL utilizada. Una verificación que se puede realizar para determinar si ha habido desbordamiento es comprobar si al sumar dos números positivos se obtiene un número negativo (*overflow*) o viceversa (*underflow*).

- *Realización de multiplicaciones*

El producto de dos variables de tamaño WL, da como resultado una variable de tamaño suma de los WL de cada variable. Siendo la longitud del IWL y del FWL de la variable resultante la suma de las longitudes de los IWL y FWL de cada una de las variables.

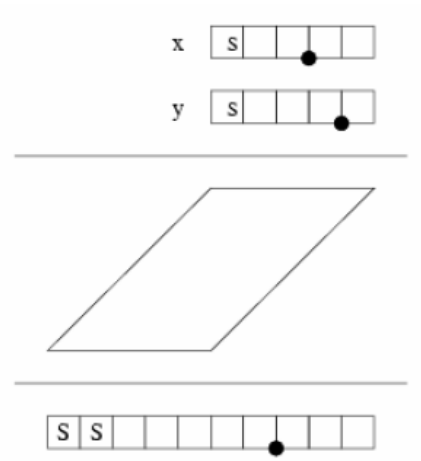


Imagen 4-4. Multiplicación de variables en punto fijo

Si suponemos que tenemos un short *x* expresado en Q5 y un short *y* expresado en Q9, el resultado de la multiplicacion de ambos es un *int* expresado en Q14. Si se quiere mantener el tamaño de las variables con el que se está trabajando sería necesario realizar una operación de

casting a la variable resultado. En un paso previo se realiza un desplazamiento para la seleccionar la Q de la variable.

$$\left.\begin{array}{l} x \;\; short \;\; Q4 \\ y \;\; short \;\; Q8 \end{array}\right\} \Rightarrow x \cdot y = z, \quad z \;\; \text{int} \;\; Q12$$
$$z' = (short)\, z >> 7, \quad z' \;\; short \;\; Q5$$

Al realizar la multiplicación y el posterior escalado se puede producir desbordamiento, por lo que es necesario tener cuidado con el valor que tienen las variables empleadas.

4.3 Principales dificultades

A la hora de portar el Verificador a punto fijo se ha tenido en cuenta el tipo de operaciones que realiza la versión en punto flotante, para realizar la adaptación de la forma más eficiente posible. Cuando se plantea realizar el portado se deben hacer un par de reflexiones previas al trabajo de programación.

- *Longitud de palabra*

Por un lado se debe elegir la longitud de palabra que se va a emplear para representar las variables. Un tamaño corto permite representar un menor rango de valores y aumenta la posibilidad de que se produzcan desbordamientos, pero tiene la ventaja de que los modelos empleados y ficheros de voz ocupan un menor tamaño. A esto hay que sumar el hecho de que hay dispositivos móviles que presentan limitaciones para trabajar con anchos de palabras superiores a 32 bits.

Una posible solución para reducir la posibilidad de desbordamiento es emplear una resolución Q diferente en función de la magnitud de las operaciones que se estén realizando en cada momento. Lo que añade una doble complejidad ya que por un lado se aumentan el número de instrucciones del programa y por otro lado hay que definir de forma precisa los rangos de variación de las variables en cada una de las etapas de la aplicación.

Dado que los terminales móviles para los que se va a implementar la aplicación no presentan limitación importante de memoria y que pueden trabajar con datos de 32 bits, se decidió adoptar este tamaño para la versión en punto fijo del Verificador.

- *Resolución*

La segunda reflexión que se debe realizar es el valor de la resolución Q y si ésta se mantendrá fija o será variable. En la selección del valor de Q, los argumentos que deben primar son ajustar el rango representable al rango de variación de las variables y reducir en la medida de lo posible la probabilidad de desbordamiento. En el presente Proyecto se tuvo que optar por una Q variable dependiendo del rango de variación de las variables a representar.

La Q elegida es variable únicamente en funciones internas dentro de una misma aplicación, es decir, en ocasiones se decide aumentar o disminuir la Q para poder efectuar la operación sin perder precisión. Sin embargo, la resolución elegida para que las diferentes aplicaciones funcionen entre sí ha sido fija (Q=7). Si se quiere modificar este valor de Q habrá que realizarlo en las diferentes aplicaciones para que todos lean y escriban lo mismo.

4.3.1 Transformación de divisiones en multiplicaciones

La operación de división es altamente ineficiente en punto fijo. Por ello se debe evitar en la medida de lo posible el empleo de la división. Una técnica empleada para solventar este problema es convertir las divisiones en multiplicaciones. La conversión consiste en transformar $a \div b$ en $a \times b^{-1}$.

Si se observa la fórmula empleada para el cálculo de las verosimilitudes para una muestra y para una Gaussiana del modelo, se puede ver que se realizan operaciones de división en dos puntos:

$$p_i\left(x_j\right) = \frac{1}{\left(2\pi\right)^{D/2}\sqrt{\prod_{k=1}^{D}\sigma_{ik}^2}} \exp\left\{-\frac{1}{2}\sum_{k=1}^{D}\frac{\left(x_{jk}-\mu_{ik}\right)^2}{\sigma_{ik}}\right\} \qquad 1 \le \mathrm{j} \le \mathrm{N}$$

Para eliminar la división dentro de la exponencial, se han modificado los modelos de locutor para que guarden σ^{-1} en lugar de σ.

Si se observa el término que precede a la exponencial, se puede ver que no depende de la señal de voz, únicamente del modelo, por lo que puede ser computado previamente y guardado como una variable del mismo (denominada C_{0i}).

Con estos cambios se consigue eliminar las divisiones y la raíz cuadrada, otra operación costosa en punto fijo. Además se reduce la carga computacional relacionada con el cálculo del término que se ha precalculado.

De esta forma se obtiene una expresión intermedia, en la que queda por solventar el cálculo de la exponencial:

$$p_i\left(x_j\right) = C_{0i} \times \exp\left\{-0.5\sum_{k=1}^{D}\left[\left(x_{jk}-\mu_{ik}\right)^2 \times \sigma_{ik}^{-1}\right]\right\} \tag{4.5}$$

4.3.2 Cálculo del logaritmo neperiano

La exponencial es también difícil de implementar en punto fijo. Si se opera en la expresión completa que proporciona el logaritmo de la verosimilitud de un fichero de voz, valor utilizado para realizar la comparación de los modelos, se llega a la siguiente expresión:

$$\log P\left(x \mid \lambda\right) = \sum_{j=1}^{N}\log\left(\sum_{i=1}^{M} w_i \times C_{0i} \times \exp\left\{-0.5\sum_{k=1}^{D}\left[\left(x_{jk}-\mu_{ik}\right)^2 \times \sigma_{ik}^{-1}\right]\right\}\right) \tag{4.6}$$

Por este camino se llega a que es necesario calcular el logaritmo de la suma de las verosimilitudes de cada muestra para cada Gaussiana. Aplicando la expresión que define el logaritmo de una suma y empleando una tabla auxiliar de logaritmos precalculados, se tiene:

$$\log_b\left(P_1+P_2\right) = \log_b\left(P_1\right) + \log_b\left(1+b^{\left(\log_b\left(P_2\right)-\log_b\left(P_1\right)\right)}\right) \tag{4.7}$$

$$tablaViterbi\left[k\right] = \log_b\left(1+b^k\right) \tag{4.8}$$

De donde se saca:

$$\log_b(P_1+P_2)=\log_b(P_1)+tablaViterbi\left[\log_b(P_2)-\log_b(P_1)\right] \quad (4.9)$$

Escogiendo *b* de forma adecuada, se consigue una simplificación en el paso de aritmética Q0 a aritmética Qn:

$$\frac{1}{\log(b)}=-2^n \quad \Rightarrow \quad \log_b(P)\ \ en\ \ Q0=-\log(P)\ \ en\ \ Qn \quad (4.10)$$

Aplicando lo explicado anteriormente se obtiene el valor de la suma de logaritmos de manera eficiente para punto fijo. Para cada vuelta del bucle que realiza el cálculo de la verosimilitud, para cada muestra de voz y para cada Gaussiana del modelo se realizan las siguientes operaciones:

Punto fijo

```
(…)                                                    //"suma_mundo" contiene el valor de
                                                       //parte del interior de la exponencial
                                                       //(Σk[(xjk-μik)^2×σik^-1])

aux = -((int)((mundo->C0[i]+suma_mundo)>>1));          //en "aux" se guarda el valor de la
                                                       //aportación (en logaritmo) de cada
                                                       //Gaussiana en Q7

aux = aux + mundo->coefMix[i];                         //se multiplica (suma de logaritmos) por
                                                       //el peso wi en Q7

aux = -aux;                                            //Cambiando de signo pasamos a Q0 en
                                                       //base b (b tal que 1/log(b) = -2^7)

if (aux2_mundo == 0 && i==0)
{
aux2_mundo=aux;                                        //se guarda el primer valor de "aux" en
                                                       //"aux2_mundo"
}
else if(aux < aux2_mundo)
{
        if (aux2_mundo-aux >= tamTablaViterbi)         //Si k se sale de la tabla, es que su
                                                       //valor asociado es 0
                aux2_mundo = aux;
        else
                aux2_mundo = aux + tablaViterbi[aux2_mundo-aux];
}
else
{
        if (aux - aux2_mundo < tamTablaViterbi)
                aux2_mundo = aux2_mundo + tablaViterbi[aux-aux2_mundo];
}
//en "aux2_mundo" va quedando almacenado el logaritmo de la suma
```

Una vez realizados los cálculos para todas las Gaussianas el valor de *aux2_mundo* contiene la aportación en forma logarítmica de esa muestra al cálculo de la verosimilitud:

```
Lprob_mundo = Lprob_mundo - aux2_mundo;        //Cambiando el signo a "aux2_mundo" se vuelve a
                                               //pasar a Q7. De esta manera se guarda en
                                               //"Lprob_mundo" el valor total de la
                                               //verosimilitud en Q7 para todas las muestras y
                                               //todas las Gaussianas.
```

Este cálculo es mucho más sencillo e intuitivo en la versión en punto flotante, donde se puede calcular la aportación de cada Gaussiana en números reales y, a continuación, realizar el cálculo del logaritmo siguiendo los pasos de las ecuaciones (3.10), (3.8) y (3.12):

Punto flotante

```
//para cada Gaussiana y cada muestra de voz
for(j=0;j<mundo->VecSize;j++)
{//Calculamos el sumatorio y el productorio de la fórmula (3.10) de la función de densidad
   sumaMundo = sumaMundo + pow((parhtk_st->param[k][j]-mundo->mean[i][j]),2)/mundo->variance[i][j];
   productorio = productorio*sqrt(2*PI*mundo->variance[i][j]);
}
//calculamos la aportación de todas las Gaussianas en conjunto (fórmula (3.8))
//este valor equivaldría al anterior "aux2_mundo" pero en unidades naturales.
auxProbM = auxProbM + mundo->coefMix[i]*exp(-0.5*sumaMundo)/productorio;
```

Una vez realizados los cálculos para todas las Gaussianas, se puede calcular el logaritmo:

```
//Por último se calcula el logaritmo a la vez que se va almacenando la suma en una variable, todo
//según indica la fórmula (3.12)
Lprob_mundo = Lprob_mundo + log(auxProbM);
```

Como se puede apreciar, la versión en punto flotante es mucho más sencilla de implementar pero dadas las características de nuestro dispositivo portátil es necesario utilizar la aritmética en punto fijo. De esta manera, nos evitamos operaciones como divisiones, exponenciales y logaritmos a cambio de aumentar la dificultad del programador.

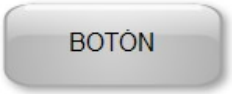
4.4 Implementación de los cuatro subsistemas

La implementación de los cuatro programas mencionados en este proyecto ha sido realizada en Visual C++ y, aunque no se muestra el código fuente completo de cada uno de los programas en el presente documento, sí se presentan a continuación los diagramas de flujo de los mismos. De esta manera, el lector puede hacerse una idea de cómo funcionan internamente cada uno de los programas empleados. En los diagramas de flujo que se muestran más adelante aparecen diversos símbolos que se explican a continuación:

Este símbolo representa el comienzo y el final del programa.

Representa un proceso interno del programa.

Con este símbolo se representa todo tipo de interfaz con el usuario, tanto nuevas ventanas como mensajes mostrados.



En gris quedan representados los diferentes botones de la aplicación.

Todo rombo dentro de los diagramas de flujos siguientes representa una decisión que el programa evalúa.

4.4.1 Grabador

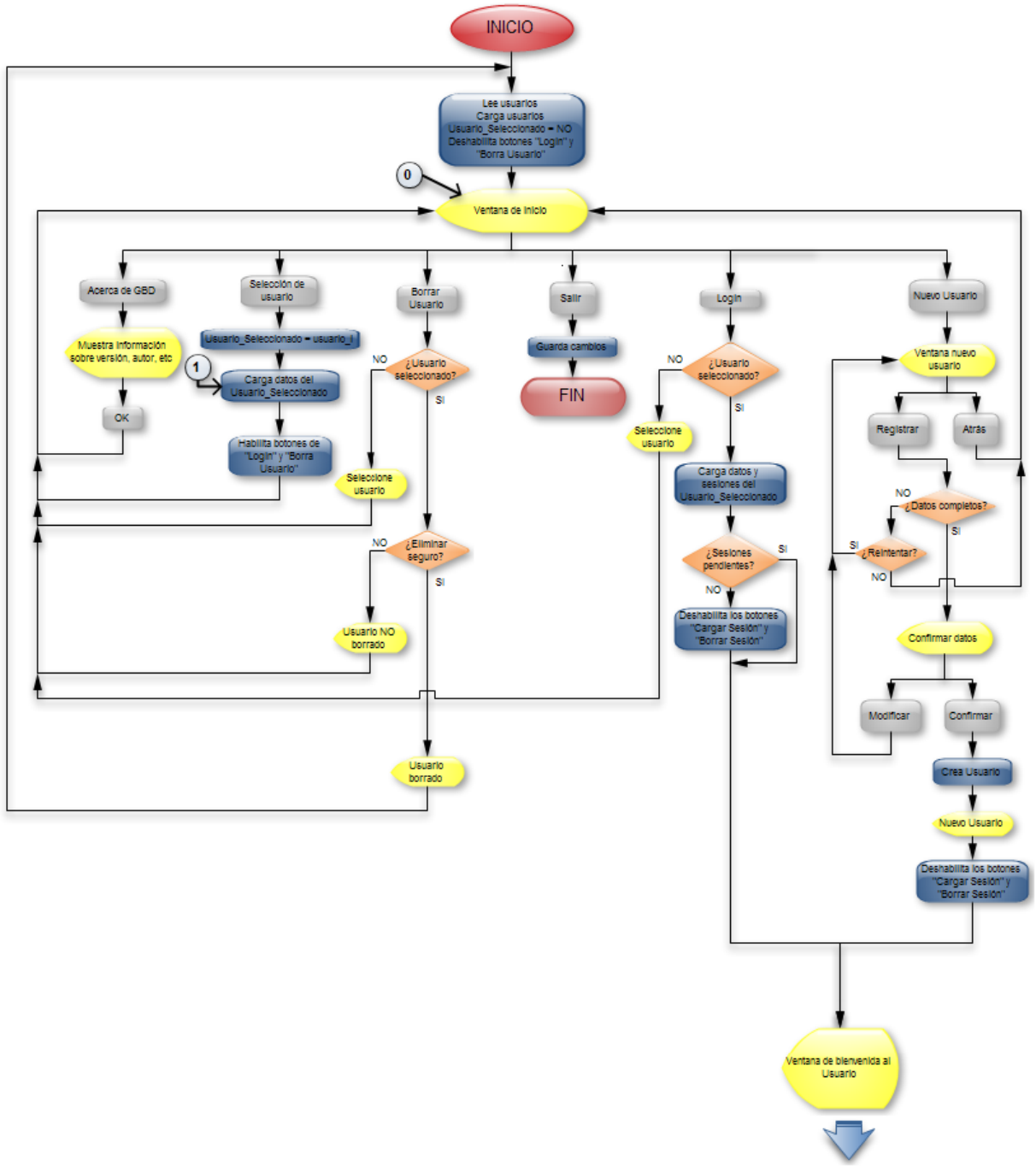


Imagen 4-5. Diagrama de flujo del Grabador (primera parte)

En el diagrama de flujo anterior se ha presentado la parte del programa hasta llegar a la pantalla de bienvenida al usuario elegido. En la siguiente imagen se muestra el diagrama de flujo a partir de la ventana comentada.

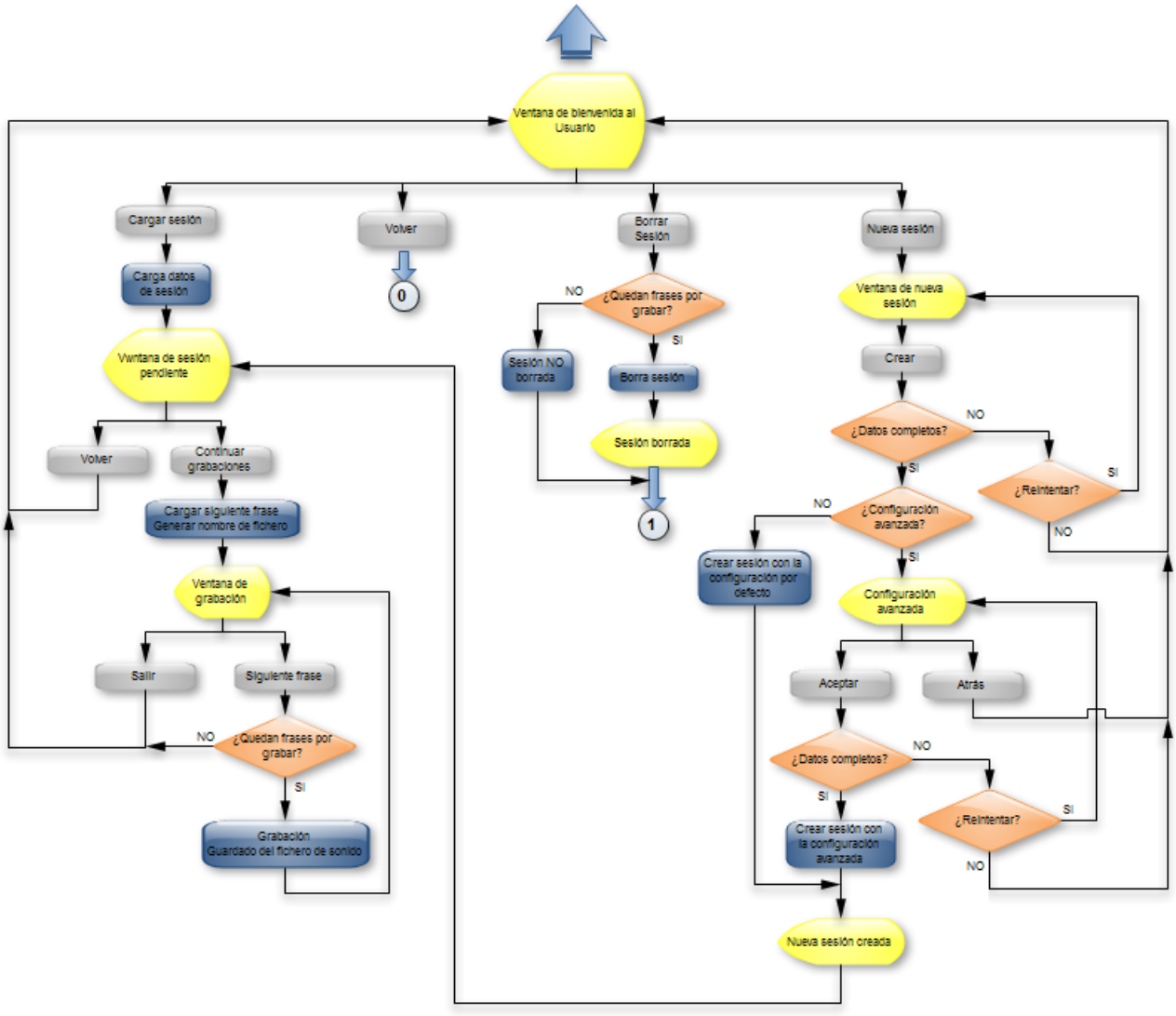


Imagen 4-6. Diagrama de flujo del Grabador (segunda parte)

4.4.2 Parametrizador

En la siguiente figura se muestra el diagrama de flujo seguido en la implementación del Parametrizador.

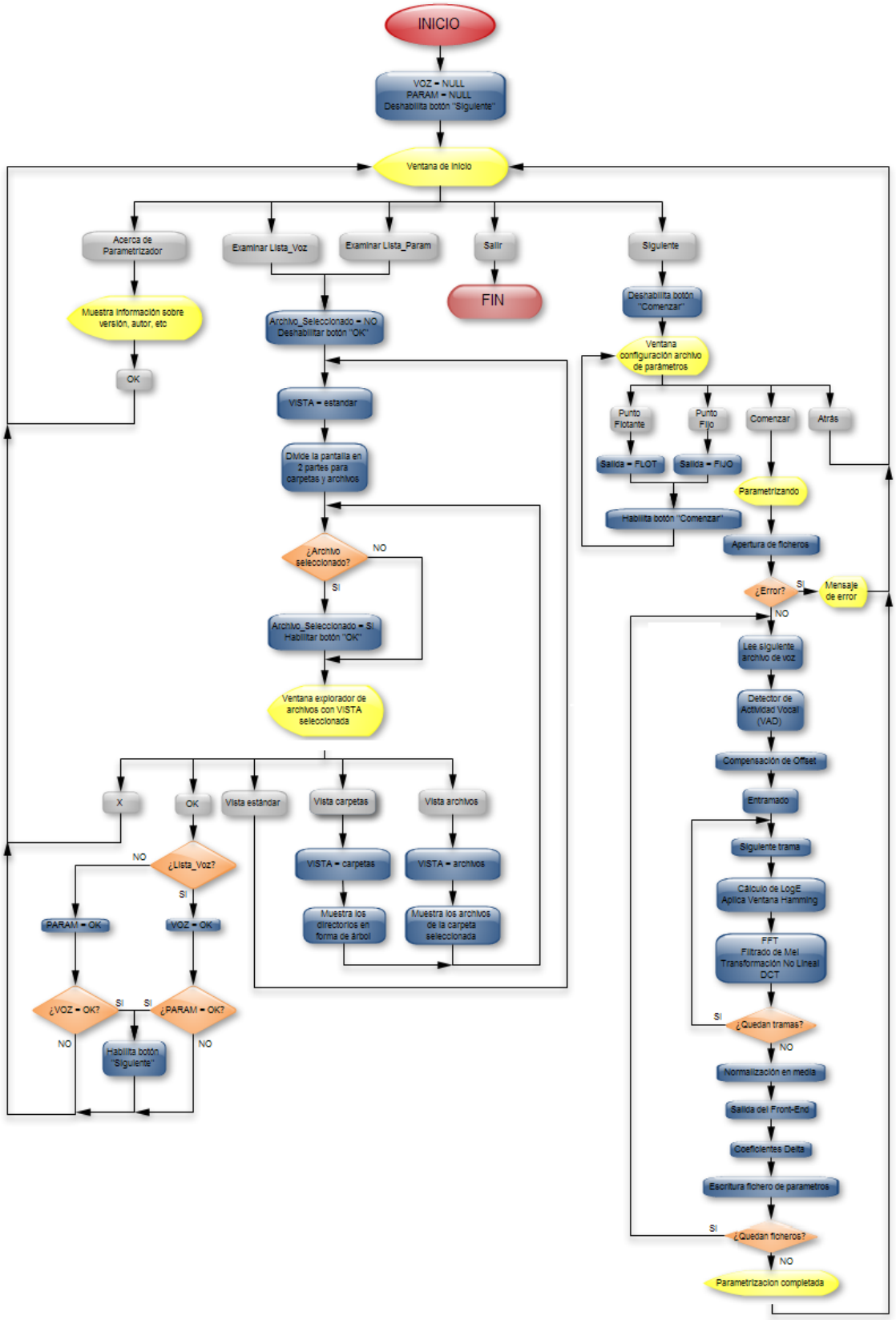


Imagen 4-7. Diagrama de flujo del Parametrizador

4.4.3 Adaptador

La siguiente imagen describe el esquema de funcionamiento del programa Adaptador realizado en la PDA.

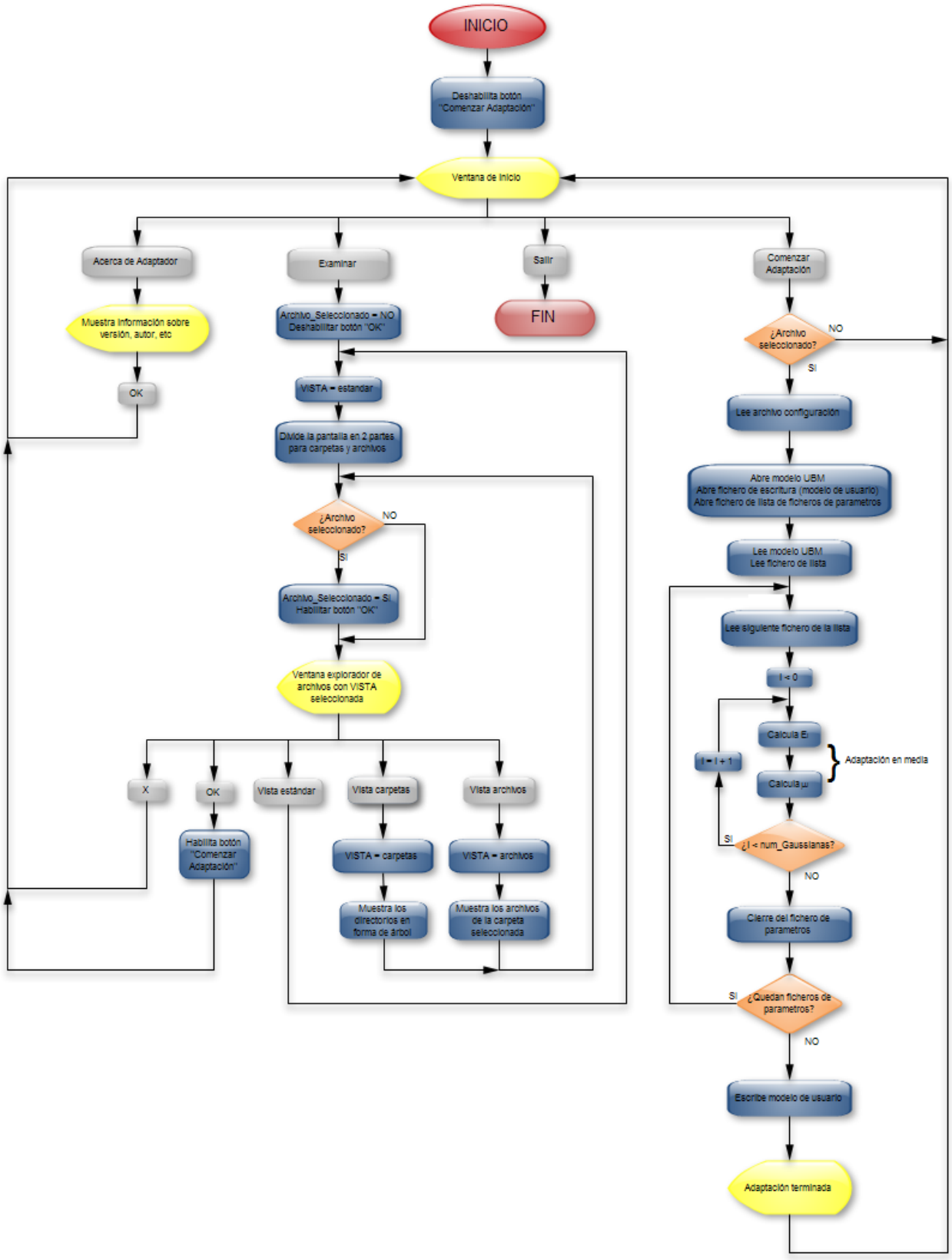


Imagen 4-8. Diagrama de flujo del Adaptador

4.4.4 Verificador

Por último, se muestra en la figura siguiente el diagrama de flujo que se ha seguido en la implementación del programa Verificador creado para la PDA en su última versión.

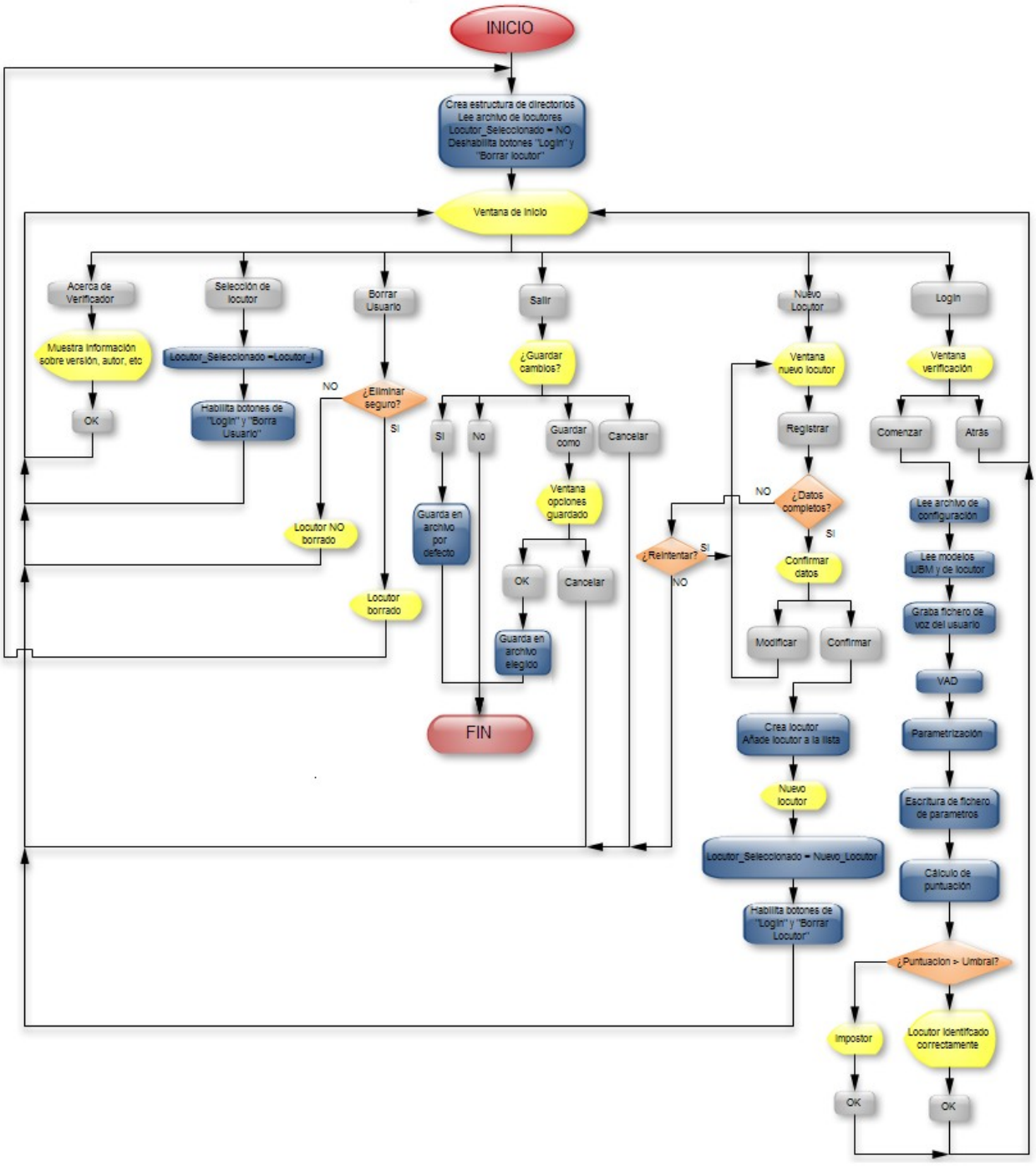


Imagen 4-9. Diagrama de flujo del Verificador

Los diagramas de flujo mostrados anteriormente pretenden que el lector comprenda el funcionamiento interno de cada uno de los programas implementados. Para el manejo de cada programa en la PDA se presenta un manual de cada aplicación en el Apéndice de este proyecto.

CAPÍTULO V

Pruebas experimentales

A lo largo del presente capítulo se explica con el mayor detalle posible todas las pruebas realizadas hasta llegar a la versión final del Verificador propuesto en este proyecto. Para ello, se comenzara explicando las condiciones de experimentación en que todas las pruebas han sido realizadas y, posteriormente, se pasará a analizar los resultados obtenidos para las diferentes versiones realizadas del programa, esto es, sus versiones para PC (en punto flotante y en punto fijo) y la definitiva para una PDA.

5.1 Condiciones de experimentación.

Como ya se ha mencionado en numerosos puntos de este documento, para poder realizar el proceso de verificación es condición necesaria el hecho de tener una base de datos con la mayor cantidad posible de ficheros de voz. De esta manera y cuanto mayor sea esta base de datos, más fiable será el modelo de mundo empleado para la adaptación de los distintos posibles usuarios del programa Verificador.

5.1.1 Base de datos empleada.

En el caso que nos ocupa en este proyecto, se ha decidido crear una base de datos propia dentro del Departamento en la que cada usuario ha grabado 3 sesiones de 50 palabras cada una. Se ha intentado, en la medida de lo posible, que las sesiones sean grabadas en días diferentes de modo que se puedan obtener todas las características posibles del usuario porque como se ha explicado en esta memoria, debido a la variabilidad intralocutor, un mismo locutor en días diferentes puede mostrar diferentes características en su voz.

5.1.1.1 Listado de palabras.

La lista de las cincuenta palabras empleadas en la grabación de cada usuario se muestra a continuación:

ABERTIS	ENDESA	REPSOL	DOS
ACCIONA	FCC	SACYR VALLEHERMOSO	TRES
ACERINOX	FERROVIAL	SCH	CUATRO
ACS	GAMESA	SOGECABLE	CINCO
ALTADIS	GAS NATURAL	TELEFONICA MOVILES	SEIS
AMADEUS	IBERDROLA	TELEFONICA	SIETE
ARCELOR	IBERIA	TERRA	OCHO
BANCO POPULAR	INDITEX	TPI	NUEVE
BANKINTER	INDRA	UNION FENOSA	SI
BBVA	METROVACESA	ZELTIA	NO
CORPORACION MAPFRE	NH HOTELES	CERO	CANCELAR
DRAGADOS	REE	UNO	AYUDA
ENAGAS	RED ELECTRICA DE ESPAÑA		

Tabla 5-1. Listado de palabras de la BBDD

5.1.1.2 Listado de locutores.

A continuación se muestran todos los locutores que han formado parte de la base de datos empleada en el momento del diseño de este proyecto. Hay que tener en cuenta que el proceso de grabación sigue su curso y con el paso del tiempo, esta base de datos será más amplia. En este punto, cabe resaltar que no todos los locutores han grabado las tres sesiones requeridas pero esto no es un impedimento ya que, aunque dichos locutores no puedan disponer de su propio modelo de locutor, sí ayudaran a crear un modelo de mundo.

Locutor	Sexo	Edad	Sesiones
abmm	mujer	29	3
afg	mujer	17	3
aga	mujer	34	3
agm	hombre	24	3
aigm	mujer	25	3
app	hombre	25	3
cds	hombre	25	3
csc	hombre	25	3
dfg	hombre	8	3
dguti	hombre	24	3
ehc	mujer	25	3
ell	hombre	25	3
emr	hombre	25	3
gcl	hombre	25	3

Locutor	Sexo	Edad	Sesiones
ibc	hombre	25	3
jfrc	hombre	26	3
jgm	mujer	41	3
jvp	hombre	25	3
mfg	mujer	15	3
mfl	hombre	25	3
mfm	hombre	45	3
mjh	mujer	29	3
mlt	hombre	31	3
rff	hombre	25	3
rmm	mujer	24	3
sgs	mujer	25	3
sht	hombre	25	3
vtg	mujer	25	3

Tabla 5-2. Locutores con modelo propio de voz

En la tabla anterior se han mostrados todos los locutores que han grabado las 3 sesiones requeridas y que, por lo tanto, poseen un modelo propio de voz con el que se han realizado las pruebas mostradas en los siguientes apartados.

Sin embargo, la base de datos posee datos de otros locutores que no han terminado de grabar por completo todas las sesiones, son los siguientes:

Locutor	Sexo	Edad	Sesiones
amn	hombre	24	2
bfr	mujer	24	2
dgj	hombre	25	2
imj	mujer	30	2
jag	hombre	26	2
jcp	hombre	25	2
jjw	hombre	25	2
jpcb	hombre	28	2
mafr	hombre	25	2
mpp	mujer	32	2
pgg	mujer	25	2
vpgj	hombre	27	2
abm	hombre	25	1
acf	hombre	32	1
alicia	mujer	28	1
bb	mujer	33	1
cpm	mujer	30	1

Locutor	Sexo	Edad	Sesiones
dcg	hombre	24	1
evm	mujer	29	1
fdm	hombre	37	1
imm	hombre	25	1
jfg	mujer	31	1
jlra	hombre	31	1
jmgc	hombre	39	1
lgv	mujer	25	1
lsa	hombre	45	1
mgm	mujer	45	1
mpg	mujer	25	1
msf	mujer	30	1
rsm	hombre	28	1
rtl	hombre	25	1
sam	hombre	24	1
sbf	hombre	25	1
vgv	mujer	24	1

Tabla 5-3. Locutores sin modelo propio de voz

5.1.1.3 Tamaño de la base de datos.

A modo de resumen de lo mostrado anteriormente, en la base de datos empleada tenemos un total de 28 usuarios que han grabado las 3 sesiones de 50 palabras y un total de 34 locutores más que no han alcanzado las 3 sesiones necesarias y que, por tanto, no poseen un modelo propio de su voz. Así pues, con todos los archivos de voz grabados se puede obtener un modelo de mundo UBM que congregará características de **6.500 ficheros de voz**.

5.1.2 Parametrización, UBM y adaptación.

Con la base de datos creada se procedió a parametrizar todos los ficheros de voz para, posteriormente emplearlos en el cálculo del modelo de mundo. En el proceso de parametrización se optó por la siguiente elección de diseño:

- Veintiséis (26) coeficientes MFCC: 12 coeficientes cepstrales más la medida de la log-energía y sus correspondientes primeras derivadas. Todos ellos normalizados en media.

Una vez se tenían todos los ficheros parametrizados de la BBDD se procedió a utilizarlos en el entrenamiento del modelo de mundo y su posterior adaptación a los diferentes modelos de usuario gracias a unos scripts implementados para ello dentro del Grupo de Procesado Multimedia. Las elecciones de diseño realizadas fueron:

- Seis mil quinientos ficheros de parámetros empleados.
- Treinta y dos Gaussianas empleadas el modelo UBM.

Una vez se obtuvo el modelo de mundo se procedió a realizar la adaptación necesaria para cada uno de los veintiocho locutores que iban a poseer un modelo propio de voz. En este proceso se utilizaron:

- Ciento cincuenta ficheros de parámetros por cada locutor.
- Treinta y dos Gaussianas para el modelo de cada locutor.

5.2 Verificador para PC.

Como ya se ha comentado durante esta memoria, antes de desarrollar la aplicación para un dispositivo móvil, tal como una PDA se decidió diseñar el verificador en un ordenador de sobremesa. El objetivo era comprobar el comportamiento de la aplicación para después proceder a su portado a la PDA.

Esta aplicación ha sido diseñada en su versión en Punto Flotante para después pasar a la versión en Punto Fijo.

5.2.1 Versión en Punto Flotante.

El primer paso que se realizó en el desarrollo de este proyecto fue utilizar la base de datos de voz grabada sobre PDA en el Departamento para después obtener los ficheros parametrizados en punto flotante. Este proceso de parametrización fue realizado gracias a una aplicación en lenguaje C++ una vez comprobados sus resultados de acuerdo a los obtenidos con el programa HTK.

El objetivo de esta prueba era determinar la Probabilidad de Error del sistema así como el Umbral de decisión.

- La Probabilidad de Error se considera igual a la Tasa Equivalente de Error (EER), es decir, el punto en donde se cruzan las curvas de FA y FR.

Con estos parámetros, los resultados obtenidos fueron los siguientes:

Usuario	PE	Umbral
Todos	18,35%	-1,76
abmm	18,46%	-2,76
afg	24,78%	-13,82
aga	13,83%	3,27
agm	7,41%	19,35
aigm	22,45%	-4,77
app	23,95%	13,32
cds	2,21%	54,02
csc	22,02%	-16,83
dfg	10,09%	24,37
dguti	11,68%	14,32
ehc	28,54%	-33,92
ell	23,67%	-32,91
emr	12,36%	32,41
gcl	8,44%	49,50
ibc	29,96%	0,25
jfrc	19,30%	9,30
jgm	16,62%	-1,76
jvp	24,77%	-26,88
mfg	20,76%	10,30
mfl	12,58%	-14,82

Usuario	PE	Umbral
mfm	19,90%	0,25
mjh	24,41%	-81,16
mlt	15,67%	20,35
rff	21,98%	10,30
rmm	7,66%	35,43
sgs	18,92%	7,29
sht	22,81%	-25,88
vtg	16,06%	-11,81
Promedio	**17,90%**	

Tabla 5-4. Probabilidad de Error en Punto Flotante para PC

En la tabla se observa el umbral óptimo obtenido para cada usuario por separado así como la PE obtenida para dicho umbral óptimo. Por otra parte se puede ver también en la misma tabla de resultados, en el usuario denominado "Todos", la PE y el umbral óptimo obtenido en caso de meter a todos los usuarios conjuntamente en el análisis. Los umbrales óptimos pueden ser muy diferentes en función del usuario y al juntar a todos los usuarios se obtiene un único umbral para todos. Esto influye en que algunos usuarios estarán más o menos cerca de su umbral óptimo (y por tanto, también de su PE óptima) pero otros estarán lejos o incluso muy lejos, con lo que la PE de estos usuarios bajaría mucho.

Es fácil ver que con esta manera conjunta de obtener un mismo umbral para todos los usuarios, casi ninguno de los usuarios se encuentra en su situación óptima. Sin embargo, se tiene la ventaja de que el umbral no depende del usuario que esté utilizando el Verificador.

En este proyecto se ha decidido que esta no es una buena solución, ya que se considera que es mejor tratar de estar en la situación óptima según el usuario. Esto implicaría que el propietario del Verificador tuviera que enviar los resultados de sus grabaciones para que los diseñadores del Verificador calcularan su umbral óptimo. Este procedimiento tiene un inconveniente con el usuario final ya que el proceso tiene un paso intermedio de comunicación, pero a cambio se asegura que el Verificador le va a funcionar de una forma óptima y adaptada a sus características personales.

Por estos motivos, se ha decidido que la PE total del Verificador es la media de todas las PE óptimas de cada usuario. Así pues, la **Probabilidad de Error** del Verificador en Punto Flotante para PC se sitúa en **17,90%.**

Esta probabilidad de error varía en función del usuario llegando a alcanzar un valor mínimo del 2,21% y un valor máximo de 29,96%. Seguramente esta variabilidad depende de las grabaciones de las 3 sesiones realizadas por cada usuario. No todas ellas han sido realizadas en un mismo ambiente pudiendo darse casos de que determinados usuarios han grabado con un mayor nivel de ruido por ejemplo. Este hecho es determinante a la hora de calcular los parámetros y por lo tanto, al final resulta que unos usuarios se discriminan mejor del resto (PE bajas) y otros lo hacen mucho peor (PE altas).

5.2.2 Normalización de puntuaciones

La afinación de umbrales de decisión es muy problemática en la verificación de locutor. Si la opción de su valor numérico es aún una cuestión abierta en la esfera (por lo general es fijado empíricamente), su fiabilidad no puede asegurarse para cualquier condición de funcionamiento. Esta incertidumbre es principalmente debido a la variabilidad de las puntuaciones entre distintos procesos de verificación.

Esta variabilidad de puntuaciones proviene de fuentes diferentes. En primer lugar, la naturaleza del material de entrenamiento puede variar entre los locutores. La diferencia también puede venir del contenido fonético, la duración, el ruido de ambiente, así como la calidad de la formación de modelo de locutor. En segundo lugar, la falta de armonía posible entre los datos de entrenamiento (usado para el modelado de locutor) y datos de prueba es el problema restante principal en el reconocimiento de locutor. Dos factores principales pueden contribuir a esta falta de armonía:

- El locutor mismo por la **variabilidad intralocutor** (variación de la voz del locutor debido a emoción, estado de salud y edad) y algunas variaciones de las condiciones ambientales de grabación.
- La **variabilidad interlocutor** (variación de voces entre locutores), que es una cuestión particular en caso del sistema basado en umbrales independientes de locutor, también tiene que considerarse como un factor potencial que afecta la fiabilidad de los umbrales de decisión.

Finalmente, en cuanto al material de entrenamiento, la naturaleza y la calidad de los segmentos de voz influyen en el valor de las puntuaciones para procesos de impostor y cliente.

La normalización de puntuaciones ha sido introducida explícitamente para enfrentarse con esta variabilidad y afinar el umbral de decisión independiente del locutor más fácilmente.

5.2.2.1 Normalización Znorm

La normalización *Znorm* es una abreviatura del inglés *"Zero Normalization"*. Esta técnica ha sido masivamente empleada en la verificación de locutor en la mitad de los años noventa. En la práctica, un modelo de locutor se prueba contra un conjunto de locuciones producidas por algún impostor, causando una distribución de puntuaciones de impostor. La media y varianza dependientes del modelo de locutor -parámetros de normalización- se estiman de esta distribución y son aplicadas como se muestra a continuación por el sistema de verificación de locutor mientras éste se encuentra funcionando:

$$\tilde{L}_\lambda(X) = \frac{L_\lambda(X) - \mu_\lambda}{\sigma_\lambda} \tag{4.11}$$

Una de las ventajas de la normalización *Znorm* es que la estimación de los parámetros de normalización puede ser realizada de forma "offline" durante el entrenamiento del modelo de locutor.

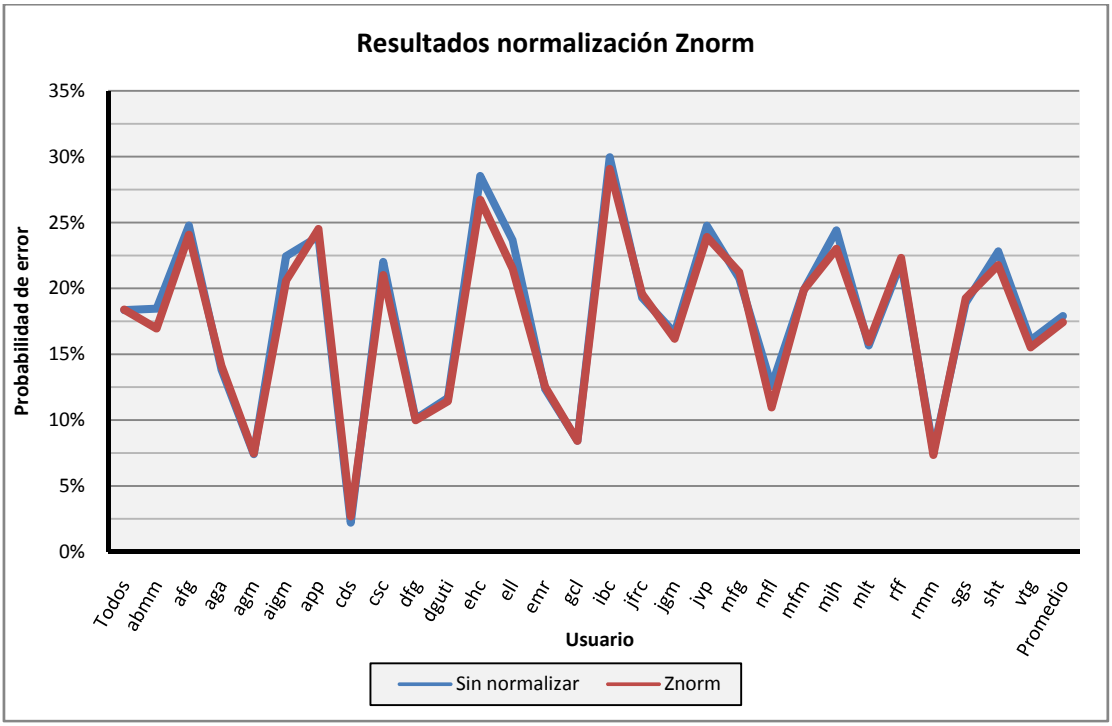
Una vez realizado este proceso de normalización se obtuvieron los siguientes resultados que se muestran en comparación con los resultados del apartado anterior con puntuaciones no normalizadas:

	Sin normalizar		Znorm	
Usuario	**PE**	**Umbral**	**PE**	**Umbral**
Todos	18,35%	-1,76	18,39%	0,89
abmm	18,46%	-2,76	16,95%	0,89
afg	24,78%	-13,82	24,07%	0,68
aga	13,83%	3,27	14,15%	0,99
agm	7,41%	19,35	7,47%	1,41
aigm	22,45%	-4,77	20,59%	0,78
app	23,95%	13,32	24,51%	0,68
cds	2,21%	54,02	2,69%	2,24
csc	22,02%	-16,83	21,00%	0,68

	Sin normalizar		Znorm	
Usuario	**PE**	**Umbral**	**PE**	**Umbral**
dfg	10,09%	24,37	9,99%	0,99
dguti	11,68%	14,32	11,44%	1,20
ehc	28,54%	-33,92	26,71%	0,47
ell	23,67%	-32,91	21,49%	0,68
emr	12,36%	32,41	12,55%	1,09
gcl	8,44%	49,50	8,42%	1,30
ibc	29,96%	0,25	29,06%	0,57
jfrc	19,30%	9,30	19,58%	0,78
jgm	16,62%	-1,76	16,17%	0,89
jvp	24,77%	-26,88	23,90%	0,57
mfg	20,76%	10,30	21,23%	0,78
mfl	12,58%	-14,82	10,97%	1,20
mfm	19,90%	0,25	19,91%	0,89
mjh	24,41%	-81,16	23,00%	0,57
mlt	15,67%	20,35	15,92%	0,89
rff	21,98%	10,30	22,31%	0,78
rmm	7,66%	35,43	7,35%	1,30
sgs	18,92%	7,29	19,25%	0,89
sht	22,81%	-25,88	21,76%	0,68
vtg	16,06%	-11,81	15,52%	0,89
Promedio	**17,90%**		**17,43%**	

Tabla 5-5. Resultados normalizando con Znorm

Si analizamos gráficamente las Probabilidades de Error en ambos casos obtenemos:



Gráfica 5-1. Probabilidad de Error aplicando Znorm y sin normalizar

Según los datos mostrados en la Tabla 5-5 y en la Gráfica 5-1 se puede apreciar que apenas se observa una leve mejora en lo que al valor de la Probabilidad de Error se refiere pasando ésta de un valor de 17,90% a un 17,43%. Esta mejora no es comparable al gasto computacional

empleado para aplicar la normalización *Znorm* por lo tanto después de realizar esta prueba se tomó la decisión de no aplicar esta técnica de normalización sobre las puntuaciones obtenidas.

5.2.2.2 Normalización Tnorm

Esta técnica sigue basada en la estimación de los parámetros media y varianza para normalizar la distribución de puntuación de impostor; la *"Test Normalization"* (*Tnorm*) se diferencia de *Znorm* por el uso de modelos de impostor en vez de locuciones de test. Durante la fase de test, la señal de voz de entrada es clásicamente comparada con el modelo de locutor afirmado así como con un juego de modelos de impostor para estimar la distribución de puntuaciones de impostor y los parámetros de normalización consecutivamente.

Si *Znorm* se considera como una técnica de normalización dependiente del locutor, *Tnorm* es una técnica dependiente de la prueba. Como la misma prueba es usada tanto durante el test como durante la estimación de parámetros de normalización, *Tnorm* evita un posible inconveniente de *Znorm* basado en una falta de armonía posible entre las condiciones de normalización y prueba. Contrariamente al caso de *Znorm*, *Tnorm* tiene que ser realizado *"online"* durante las pruebas.

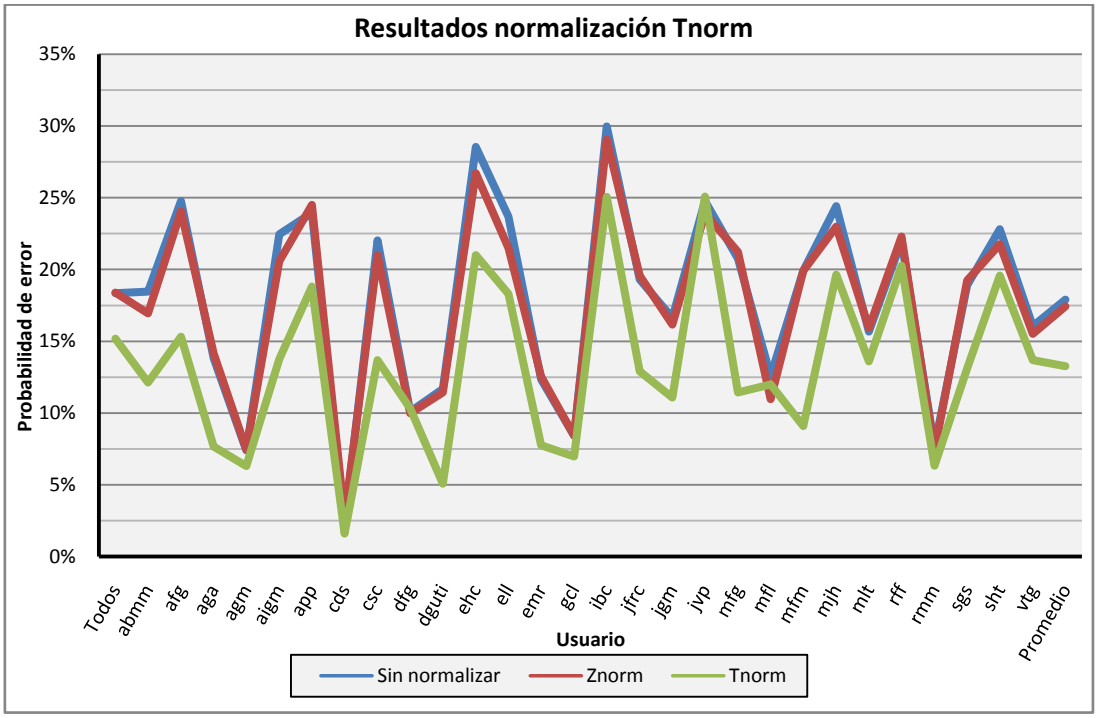
Realizando esta normalización sobre los mismos datos analizados anteriormente para el caso sin normalizar y aplicando *Znorm* obtenemos los resultados mostrados en la tabla siguiente:

	Sin normalizar		**Znorm**		**Tnorm**	
Usuario	**PE**	**Umbral**	**PE**	**Umbral**	**PE**	**Umbral**
Todos	18,35%	-1,76	18,39%	0,89	15,19%	0,97
abmm	18,46%	-2,76	16,95%	0,89	12,13%	0,82
afg	24,78%	-13,82	24,07%	0,68	15,31%	0,97
aga	13,83%	3,27	14,15%	0,99	7,66%	0,97
agm	7,41%	19,35	7,47%	1,41	6,30%	1,45
aigm	22,45%	-4,77	20,59%	0,78	13,75%	1,13
app	23,95%	13,32	24,51%	0,68	18,82%	1,13
cds	2,21%	54,02	2,69%	2,24	1,60%	1,29
csc	22,02%	-16,83	21,00%	0,68	13,69%	0,82
dfg	10,09%	24,37	9,99%	0,99	10,32%	1,45
dguti	11,68%	14,32	11,44%	1,20	5,09%	1,13
ehc	28,54%	-33,92	26,71%	0,47	21,00%	0,66
ell	23,67%	-32,91	21,49%	0,68	18,31%	0,50
emr	12,36%	32,41	12,55%	1,09	7,76%	1,45
gcl	8,44%	49,50	8,42%	1,30	6,97%	1,45
ibc	29,96%	0,25	29,06%	0,57	25,06%	1,13
jfrc	19,30%	9,30	19,58%	0,78	12,90%	1,13
jgm	16,62%	-1,76	16,17%	0,89	11,09%	0,97
jvp	24,77%	-26,88	23,90%	0,57	25,08%	0,66
mfg	20,76%	10,30	21,23%	0,78	11,44%	1,29
mfl	12,58%	-14,82	10,97%	1,20	11,99%	0,50
mfm	19,90%	0,25	19,91%	0,89	9,10%	1,13
mjh	24,41%	-81,16	23,00%	0,57	19,64%	0,03
mlt	15,67%	20,35	15,92%	0,89	13,60%	1,45
rff	21,98%	10,30	22,31%	0,78	20,27%	1,29
rmm	7,66%	35,43	7,35%	1,30	6,32%	1,13
sgs	18,92%	7,29	19,25%	0,89	13,10%	1,13
sht	22,81%	-25,88	21,76%	0,68	19,58%	0,66
vtg	16,06%	-11,81	15,52%	0,89	13,68%	0,97
Promedio	**17,90%**		**17,43%**		**13,27%**	

Tabla 5-6. Resultados con normalización Tnorm

En este caso, sí se observa una mejora de la Probabilidad de Error al aplicar la técnica de normalización *Tnorm*. La PE de cada usuario baja de media 4,63 puntos porcentuales con respecto a las PE calculadas sin aplicar normalización. Con estos resultados, la PE total se situaría en un **13,27%.**

En la siguiente gráfica mostramos los resultados obtenidos en las 3 situaciones:



Gráfica 5-2. Comparativa PE para distintas normalizaciones

Analizando gráficamente la situación observamos que el uso de esta técnica de normalización mejora la Probabilidad de Error de casi todos los usuarios. Este hecho se aprecia debido a que la curva representada con *Tnorm* sigue la misma forma de onda que las anteriores pero se encuentra desplazada verticalmente hacia unos valores más bajos.

A pesar de obtenerse mejores resultados en este punto aplicando *Tnorm*, en el presente Proyecto no se ha decidido aplicar ninguna normalización debido a la complejidad de incluir la misma en el algoritmo de verificación. Sin embargo, podría ser una línea de trabajo a seguir para futuras versiones de este Verificador dado que sus resultados parecen prometer una mejor Probabilidad de Error.

De todas maneras, todavía quedaría analizar si el uso de esta técnica sigue siendo favorable (en términos de PE) al pasar a Punto Fijo y, posteriormente, al portar el programa a la PDA ya que, como veremos a continuación, la PE empeora en cada uno de estos pasos.

5.2.3 Versión en Punto Fijo

Una vez realizada la versión en punto flotante y habiendo analizado sus resultados se procedió a transformar el programa a punto fijo para evaluar así los nuevos resultados y obtener una comparación con la versión en punto flotante.

Como ya se ha comentado, esta transformación de la aplicación a Punto Fijo no es para nada una obviedad ya que ha habido que revisar completamente el código y modificar todas aquellas funciones en las que existían operaciones muy costosas de realizar en Punto Fijo.

Los resultados obtenidos se observan en la siguiente tabla que tiene la misma estructura que las vistas hasta ahora:

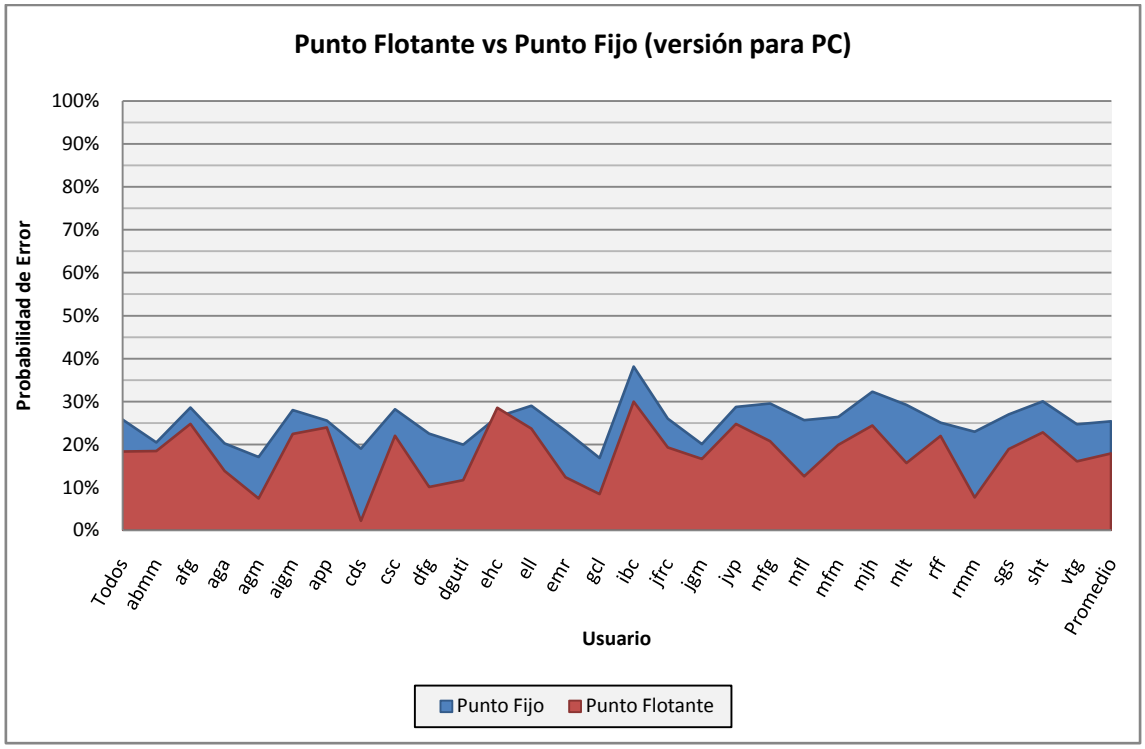
Usuario	PE	Umbral
Todos	25,83%	40,20
abmm	20,46%	3,02
afg	28,59%	21,11
aga	20,23%	37,19
agm	17,07%	56,28
aigm	28,00%	27,14
app	25,55%	53,27
cds	19,01%	118,22
csc	28,19%	44,22
dfg	22,51%	83,42
dguti	19,95%	58,29
ehc	26,42%	39,20
ell	29,00%	9,05
emr	23,17%	78,39
gcl	16,84%	84,42
ibc	38,13%	24,12
jfrc	26,00%	35,18
jgm	20,08%	60,30
jvp	28,72%	39,20
mfg	29,53%	52,26
mfl	25,65%	13,07
mfm	26,40%	35,18
mjh	32,28%	-24,12
mlt	29,21%	65,33
rff	25,07%	59,30
rmm	22,95%	90,45
sgs	27,00%	19,10
sht	30,03%	10,05
vtg	24,69%	34,17
Promedio	**25,38%**	

Tabla 5-7. Probabilidad de Error en Punto Fijo para PC

Del mismo modo que en el caso de punto flotante definimos la PE del Verificador como la media de todas las PE óptimas de cada usuario analizado. De esta manera, la **Probabilidad de Error** del Verificador en Punto Fijo para PC se sitúa en **25,38%.**

5.2.4 Comparación Punto Flotante y Punto Fijo

En los anteriores apartados hemos obtenido las probabilidades de error para las versiones en punto flotante y en punto fijo. En la siguiente gráfica podremos observar de una manera visual las diferencias entre ambas versiones:



Gráfica 5-3. Comparación PE para PC

Según los datos mostrados la Gráfica 5-3 se aprecia que la Probabilidad de Error aumenta un poco al pasar a Punto Fijo. Este hecho ya estaba previsto, debido a que, como ya se ha comentado, al pasar de Punto Flotante a Punto Fijo se pierde precisión en todos los cálculos al hacerlos más sencillos y por tanto, todos estos errores se van acumulando en una PE más elevada.

En la siguiente tabla resumimos la PE Total del Verificador para las dos versiones de PC en Punto Flotante y en Punto Fijo:

	PE
Punto Flotante	17,90%
Punto Fijo	25,38%

Tabla 5-8. Comparación entre la PE en Punto Fijo y Flotante para PC

5.3 Verificador para PDA

Una vez realizadas todas las pruebas sobre la versión para PC se procedió a portar el programa a PDA para, de nuevo, evaluar su Probabilidad de Error. En este caso solo existe la versión en Punto Fijo dadas las características del dispositivo que se está empleando. Se realizaron las mismas pruebas de la Fase de Test ya comentada en el apartado 3.1.3 de esta memoria. Con todas estas pruebas realizadas se obtuvieron los resultados expuestos en la siguiente tabla:

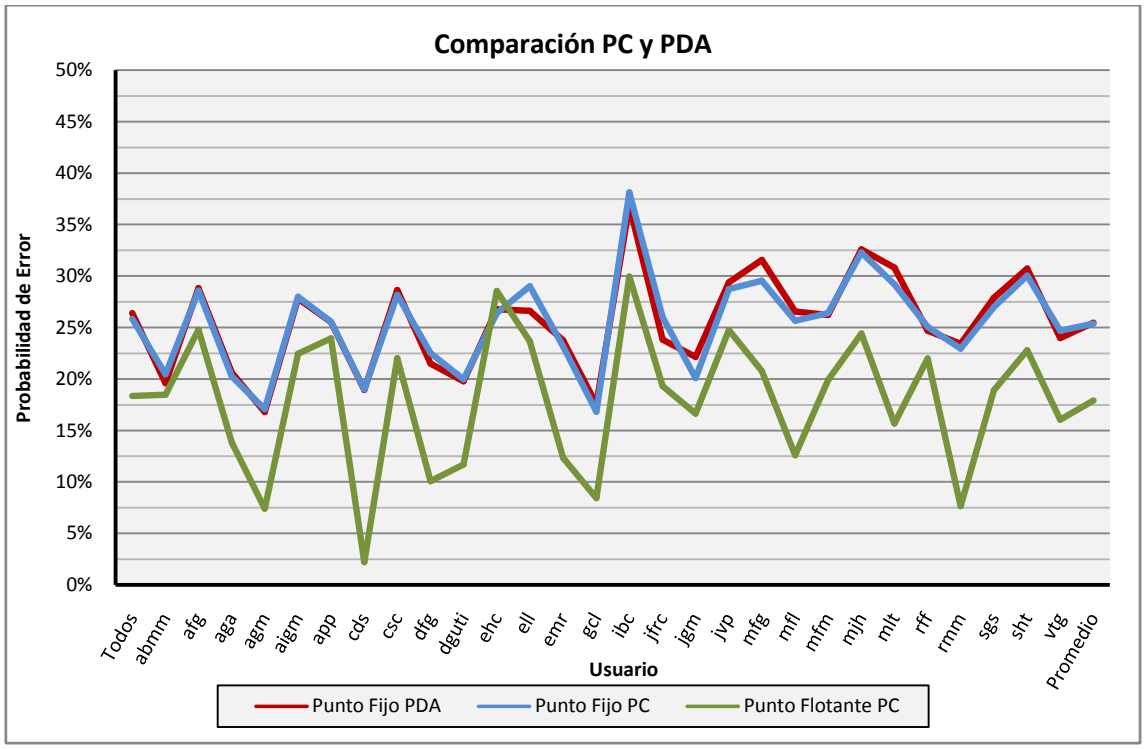
Usuario	PE	Umbral
Todos	26,40%	43,88
abmm	19,61%	7,14
afg	28,83%	19,39
aga	20,62%	43,88
agm	16,80%	56,12
aigm	27,80%	25,51
app	25,52%	50,00
cds	18,94%	98,98
csc	28,64%	43,88
dfg	21,49%	80,61
dguti	19,77%	50,00
ehc	26,78%	37,76
ell	26,61%	13,27
emr	23,76%	80,61
gcl	17,53%	86,73
ibc	36,82%	25,51
jfrc	23,82%	37,76
jgm	22,15%	74,49
jvp	29,40%	31,63
mfg	31,57%	50,00
mfl	26,55%	13,27
mfm	26,20%	37,76
mjh	32,60%	-23,47
mlt	30,82%	56,12
rff	24,71%	62,24
rmm	23,41%	86,73
sgs	27,88%	25,51
sht	30,72%	7,14
vtg	23,97%	43,88
Promedio	25,48%	

Tabla 5-9. Resultados de la versión para PDA

Siguiendo la misma lógica aplicada en los anteriores casos en Punto Flotante y Punto Fijo en sus versiones para PC, en este caso nos encontramos con una **Probabilidad de Error** en Punto Fijo para una PDA de **25,48%**.

5.4 Comparación resultados PC y PDA

En este punto podemos comparar las tres versiones comentadas, las dos ya mostradas para un ordenador de sobremesa y la versión en Punto Fijo para un dispositivo portable tal como una PDA. En la siguiente gráfica se puede ver la evolución de la PE para cada uno de los usuarios según las distintas versiones realizadas.



Gráfica 5-4. Comparación de la Probabilidad de Error entre PC y PDA

En la siguiente tabla resumimos la PE Total del Verificador para las dos versiones de PC en Punto Flotante y en Punto Fijo y para la versión de PDA:

	PE
Punto Flotante PC	17,90%
Punto Fijo PC	25,38%
Punto Fijo PDA	25,48%

Tabla 5-10. Comparación de la PE entre las versiones en PC y PDA

En este caso observamos que la Probabilidad de Error no varía sustancialmente al portar la aplicación del ordenador a la PDA. La variación en la PE ha sido debido a la transformación del programa de punto flotante a punto fijo y podemos concluir que los resultados obtenidos en PC se corresponden bastante bien con lo que finalmente han sido los resultados obtenidos en la PDA donde únicamente ha variado la PE total en una décima porcentual.

CAPÍTULO VI

Conclusiones y líneas futuras

En la presente memoria se recogen tanto los procedimientos como los resultados obtenidos durante la implementación de un verificador de hablante independiente del texto para dispositivos portables como una PDA.

El principal objetivo que motivó la realización del presente proyecto fue la obtención de una aplicación ágil para un dispositivo de capacidad limitada que proporcionara una tasa de acierto comparable a la obtenida con el paquete HTK. Para ello se eligió una tecnología madura en el ámbito del procesamiento del habla, modelos de mezcla de Gaussianas (GMM: Gaussian Mixture Models) que cumple con los requisitos fijados, ligero computacionalmente y con buenas prestaciones.

En el presente capítulo se presenta una recopilación de las tareas realizadas a lo largo de este Proyecto Fin de Carrera, así como de las principales conclusiones obtenidas y las principales dificultades encontradas. Finalmente se esbozan las principales líneas de mejora de la aplicación implementada.

6.1 Conclusiones

El objetivo final de este proyecto era crear un Verificador de locutor independiente del texto pero, para llegar al mismo se han tenido que implementar por separado diferentes programas auxiliares.

En primer lugar, hemos visto que era necesario disponer de una base de datos y en este proyecto se optó por utilizar una base de datos propia grabada con el mismo dispositivo que finalmente iba a utilizar la aplicación del Verificador. De esta manera, el proceso de verificación tendría una mayor tasa de éxito. El diseño de este **grabador para PDA** debía cumplir con una serie de características:

- Sencillo para el usuario final.
- Interfaz amigable.
- Buen resultado de grabación.

En este sentido, se ha considerado que la aplicación realizada cumple con los requisitos exigidos habiendo sido probada por diferentes usuarios que han dado opiniones favorables.

El siguiente paso en la evolución de la trayectoria del proyecto era crear un programa que realizara la **parametrización** de cada uno de los ficheros de voz de que se iban a disponer. En el capítulo 3 (apartado 3.3) se ha detallado toda la teoría con respecto al proceso de parametrización explicando cada una de las partes de que consta este proceso. En primer lugar se comenzó implementando esta aplicación en una versión para PC y después se procedió a su portado a un dispositivo PDA, con las dificultades ya comentadas (ver capítulo 4) debidas a las diferentes características de dicho dispositivo.

Este módulo de código ya había sido desarrollado en estudios previos realizados por el Grupo de Procesado Multimedia del departamento de Teoría de la Señal y Comunicaciones de la Universidad Carlos III de Madrid. Sin embargo, hubo de ser modificado para adaptarse a la nueva plataforma y nuevas necesidades encontradas (normalización en media de los parámetros MFCC, apartado 5.1.2). Este programa hubo de realizarse en dos versiones, una en punto flotante y otra en punto fijo. El hecho de comenzar en punto flotante no fue debido a otro motivo que al de comenzar utilizando un código mucho más amigable para el programador, de otro modo, haber comenzado a programar en punto fijo directamente hubiera sido mucho más complicado y no se hubieran obtenido los mismos resultados. Por otro lado, el hecho de haber creado la versión para punto flotante y punto fijo nos dio la posibilidad de comparar ambas versiones en la posterior utilización del verificador final para PC (capítulo 5, apartado 5.2.4).

Una vez parametrizados todos los ficheros, se procedió al entrenamiento del modelo de mundo. Este paso no se ha podido realizar en la PDA debido a las limitadas características de este dispositivo y no hubo más remedio que realizarlo en un ordenador de sobremesa. Este paso se hizo utilizando unos scripts del Departamento que simulan el comportamiento de la herramienta HTK y con resultados satisfactorios ya probados dentro del Departamento.

Con el modelo de mundo UBM creado había que **adaptarlo** a cada uno de los locutores de la base de datos que son posibles usuarios del sistema Verificador final. Para este proceso se creó una nueva aplicación en la PDA, la cual recibe el modelo de mundo y los ficheros del usuario que se quiere adaptar y da como resultado el modelo del usuario deseado. Como siempre en este proyecto, se pensó que realizar una aplicación con una interfaz de usuario agradable de utilizar era una necesidad más dentro de los objetivos propuestos al comenzar el proyecto. El proceso de Adaptación, a diferencia del de entrenamiento del UBM, sí ha podido ser realizado en un dispositivo tan limitado como una PDA ya que la cantidad de operaciones ejecutadas y la capacidad requerida del dispositivo es considerablemente menor en este caso. En la implementación de este Adaptador se ha utilizado punto por punto, el algoritmo de adaptación comentado en el capítulo 3 (apartado 3.4.3) con la excepción de que en este proyecto se decidió realizar la adaptación solo en media y dejar fija la varianza debido a que

los resultados obtenidos se consideraron lo suficientemente satisfactorios como para no complicar demasiado la programación y no saturar en exceso las limitaciones de la PDA.

Con todos estos programas ya creados, se implementó el **Verificador** final que tiene partes de algunos de ellos, ya que esta aplicación debe grabar la voz al locutor, debe realizar la parametrización y por último, realizar la verificación en sí.

En el capítulo 5 se muestran los resultados más significativos obtenidos dentro de la realización del presente proyecto. En él se puede ver que con la versión realizada en punto flotante para PC se obtuvieron mejores resultados en cuanto a PE que en la posterior versión en punto fijo (apartado 5.2.4). Este hecho, era más que esperado, ya que al tener que realizar las operaciones en punto fijo siempre se pierde información que, a la postre, da lugar a una peor Probabilidad de Error del sistema. Antes de comparar con la versión en punto fijo, se estudió la posibilidad de realizar una normalización de puntuaciones, analizando para ello los resultados obtenidos con las diferentes normalizaciones *Znorm* y *Tnorm*. Efectivamente, se obtuvieron mejores resultados (apartado 5.2.2), sobre todo con *Tnorm* pero no era el objetivo de este proyecto continuar por esa vía, en este sentido, se deja una línea de futuro en la cual se podría comenzar a trabajar. El último paso en todo el proceso empleado en este trabajo fue realizar el portado de la aplicación a la PDA, no fue tarea fácil, ya que, como se ha explicado, había que modificar el código debido a que el lenguaje utilizado para el programa en PC y PDA no era el mismo. Al cambiar de plataforma, se esperaba obtener resultados inferiores y sin embargo, los resultados obtenidos en la versión de la PDA son muy similares a los obtenidos para la versión en punto fijo para PC (apartado 5.4) de modo que podemos concluir que los resultados finales obtenidos se ajustan en todo momento a lo que en principio esperábamos.

No hay que perder de vista, que en el presente proyecto se trataba de obtener un Verificador destinado a un posible usuario final, a una posible entrada en el mercado. Esto implica que el programa realizado debía estar bien depurado, con una interfaz de usuario sencilla y amigable y no únicamente un programa de laboratorio. Esta parte ha sido muy importante a la hora de realizar el proyecto y se ha logrado obtener un producto final que podría ser introducido en el mercado.

6.2 Líneas futuras

En este trabajo nos hemos centrado en la realización de un Verificador de locutor independiente de texto que tuviera la menor Probabilidad de Error posible. Esto se ha conseguido pero sin embargo consideramos que existen nuevos caminos que se pueden todavía explorar para conseguir mejores prestaciones.

En el proceso de creación de la base de datos se podría utilizar un diccionario más extenso de modo que queden recogidas más características de la voz. En busca del mismo objetivo, también se podría aumentar el número de sesiones necesarias para poder obtener un modelo propio. Todo ello influiría en un mejor cálculo del modelo de mundo y, sobre todo, en una mejor adaptación de éste a cada uno de los usuarios.

Como se ha comentado en el apartado anterior, el proceso de entrenamiento del modelo de mundo ha de ser realizado fuera del dispositivo portable. Esto no tiene por qué ser un inconveniente ya que el usuario final no debe formar parte del entrenamiento del modelo UBM sino que éste se le proporciona, pero podría ser interesante lograr portar esta aplicación también a la PDA. En estos momentos es muy complicado, ya que con los dispositivos actuales no se posee la capacidad necesaria para realizar tal cantidad de operaciones y un ordenador de sobremesa es mucho más eficiente a día de hoy.

En este proyecto se ha optado por realizar la adaptación en media únicamente, de modo que no se aplica completamente el algoritmo de adaptación, que incluye también la posibilidad de adaptar las varianzas. En el caso que nos ocupa, se decidió adaptar únicamente las medias por no saturar demasiado a la PDA, pero en un futuro se podría estudiar si los resultados obtenidos adaptando medias y varianzas mejoran lo suficiente como para considerar su implementación.

Una última línea sobre la que se puede seguir trabajando en un futuro es la de la normalización de puntuaciones. Como se vio en el apartado 5.2.2, la normalización *Tnorm* obtiene mejores prestaciones por lo que podría ser conveniente su utilización en un futuro.

Evidentemente, el autor de este documento no es un experto programador por lo que toda revisión del código podría dar lugar a una optimización del mismo y de esta manera, se podrían obtener mejores prestaciones del sistema final, ya sea en recursos utilizados, tiempo empleado, etc.

CAPÍTULO VII

Presupuesto

7.1 Presupuesto

Una parte fundamental a la hora de desarrollar un proyecto de ingeniería es hacer una estimación de los costes económicos que supondría su desarrollo. Estos van a marcar la viabilidad económica del mismo y muy seguramente determinarán la realización o no del proyecto.

En el presente capítulo se presenta un cálculo aproximado del presupuesto, en el que se han desglosado las partidas en los siguientes apartados

- Presupuesto de Ejecución Material
- Gastos generales y beneficio industrial
- Honorarios por la redacción y dirección del proyecto
- Presupuesto total

El Presupuesto de Ejecución Material junto a los gastos generales y el beneficio industrial constituyen lo que se denomina como Presupuesto de Ejecución por Contrata.

Éste, junto con los honorarios por la redacción y dirección del proyecto, dan lugar al presupuesto total.

7.1.1 Presupuesto de Ejecución Material

El presupuesto de ejecución material recoge los costes de los recursos empleados (*hardware*, *software* y otros) así como la mano de obra.

7.1.1.1 Costes de mano de obra

Para la ejecución del presente proyecto ha sido necesario un ingeniero de telecomunicación encargado de diseñar y desarrollar el proyecto y un administrativo para las labores de redacción, presentación y encuadernación del proyecto.

7.1.1.1.1 Sueldo base

	Sueldo Base Mensual	Sueldo Base Diario
Ingeniero	2.100 €	70 €
Administrativo	900 €	30 €

Tabla 7-1. Sueldo base

7.1.1.1.2 Relación de obligaciones sociales

A los sueldos base de cada uno de los trabajadores hay que añadir los costes procedentes de las obligaciones sociales correspondientes:

Contingencias comunes	23,60%
Desempleo y formación	7,00%
Accidentes de trabajo y enfermedad	1,00%
Total de obligaciones sociales	*31,6%*

Tabla 7-2. Relación de obligaciones sociales

7.1.1.1.3 Salarios efectivos

Añadiendo las obligaciones sociales a los sueldos base del ingeniero y el administrativo se obtienen los salarios efectivos de ambos:

	Sueldo Base Diario	Cargas Sociales	Total Diario
Ingeniero	70,00 €	22,12 €	92,12 €
Administrativo	30,00 €	9,48 €	39,48 €

Tabla 7-3. Salarios efectivos

7.1.1.1.4 Importe total de los salarios

Según el tiempo empleado en cada una de las fases de que consta el presente Proyecto, se calcula el coste estimado de los salarios que se muestra en la siguiente tabla:

Fase	Días	Salario Diario	Total
Estudio previo	30	92,12 €	2.763,60 €
Análisis y Diseño	90	92,12 €	8.290,80 €
Implementación	150	92,12 €	13.818,00 €
Pruebas experimentales	45	92,12 €	4.145,40 €
Memoria	30	92,12 €	2.763,60 €
Redacción	15	39,48 €	592,20 €
Total	*360*	-	***32.373,60 €***

Tabla 7-4. Importe total de los salarios

7.1.1.2 Coste de los recursos materiales

Englobamos en este apartado los gastos asociados al material fungible y amortización de equipos y herramientas de desarrollo.

7.1.1.2.1 Material

En este apartado se ha considerado el coste del entorno de trabajo y todos sus gastos asociados al consumo de electricidad y materiales de oficina. Se ha considerado un coste de 200 € al mes considerando un lugar de trabajo compartido entre varios ingenieros.

Material	Coste mensual	Meses	Total
Sitio de trabajo	200 €	12	2.400,00 €
Total	*200 €*	*12*	***2.400,00 €***

Tabla 7-5. Material

7.1.1.2.2 Equipo

Para la realización de este Proyecto ha sido necesario utilizar distintos equipos tales como un PC, una PDA y una impresora del departamento. El coste de estos elementos se muestra en la tabla siguiente considerando el porcentaje de amortización de los mismos.

Herramienta	Coste	Amortización	Total
Ordenador personal	1.100,00 €	20%	220,00 €
Impresora Láser	2.000,00 €	10%	200,00 €
PDA	300,00 €	33%	100,00 €
Total	*3.400,00 €*	-	***520,00 €***

Tabla 7-6. Equipo

7.1.1.2.3 Licencias software

En este proyecto no se ha utilizado ninguna aplicación especial más allá del Sistema Operativo de Microsoft Windows XP y el paquete de programas de Microsoft Office. El resto de aplicaciones utilizadas han sido software libre o gratuito.

Programa	Coste	Amortización	Total
MS Windows XP	120,00 €	20%	24,00 €
MS Office para redacción	130,00 €	10%	13,00 €
Total	*250,00 €*	-	***37,00 €***

Tabla 7-7. Licencias software

7.1.1.2.4 Importe total de recursos materiales

Realizando la suma de los diferentes conceptos obtenemos el total de los recursos materiales empleados en este Proyecto.

Concepto	Total
Material	2.400,00 €
Equipos	520,00 €
Licencias	37,00 €
Total	***2.957,00 €***

Tabla 7-8. Importe total de recursos materiales

7.1.1.3 Coste total de los recursos

La suma de los costes por mano de obra y de los costes por recursos materiales es lo que constituye el Presupuesto de Ejecución Material (P.E.M.).

Concepto	Coste
Coste de mano de obra	32.373,60 €
Coste de recursos materiales	2.957,00 €
Total	***35.330,60 €***

Tabla 7-9. Coste total de los recursos

7.1.2 Gastos generales y beneficio industrial

Bajo Gastos generales se incluyen todos aquellos gastos derivados de la utilización de instalaciones, cargas fiduciarias, amortizaciones, gastos fiscales, etc. Con esto, el presupuesto queda de la siguiente manera:

Concepto	Coste
Presupuesto de Ejecución Material	35.330,60 €
Gastos generales (13% PEM)	4.592,98 €
Beneficio industrial (6% PEM)	2.119,84 €
Total	***42.043,42 €***

Tabla 7-10. Presupuesto de Ejecución por Contrata

7.1.3 Honorarios por redacción y dirección del proyecto

Los honorarios del ingeniero se calculan como un porcentaje sobre el Presupuesto de Ejecución Material del Proyecto. Dicho porcentaje es del 7% pero se debe modificar por un coeficiente reductor que afecta a distintos tramos dentro del Presupuesto de Ejecución Material. Es decir, se divide el Presupuesto de Ejecución Material en tramos, se aplica un coeficiente reductor distinto en cada tramo, y se suma el resultado de cada tramo para obtener la cantidad final sobre la que se calculan los honorarios del ingeniero. En la tabla siguiente se resume el proceso:

Coeficiente	Tramo	Cantidad afectada	Porcentaje	Total
1	PEM < 30.050 €	30.050,00 €	7%	2.103,50 €
0.9	30.050€ < PEM < 60.101 €	11.993,42 €	7%	755,59 €
0.8	60.101€ < PEM < 90.152 €	-	7%	-

Tabla 7-11. Honorarios por redacción y dirección por tramo

Concepto	Coste
Subtotal acumulado	39.143,60 €
Honorarios por redacción	2.859,09 €
Honorarios por dirección	2.859,09 €
Total	***47.761,60 €***

Tabla 7-12. Honorarios por redacción y dirección del proyecto

7.1.4 Presupuesto total

Finalmente, aplicando el 16% de IVA, se obtiene el presupuesto total.

Concepto	Coste
Subtotal acumulado	47.761,60 €
I.V.A. (16%)	7.641,86 €
Total	***55.403,46 €***

Tabla 7-13. Presupuesto total

El presupuesto de este proyecto asciende a CINCUENTA Y CINCO MIL CUATROCIENTOS TRES EUROS CON CUARENTA Y SEIS CÉNTIMOS.

Madrid, 4 de marzo de 2010

Fdo. César Díez Sánchez

Ingeniero de Telecomunicación

APÉNDICE

Manuales de usuario

En este capítulo se expone una descripción detallada de las cuatro aplicaciones desarrolladas en el presente Proyecto de modo que el usuario final pueda utilizar cada uno de los menús creados.

8.1 Grabador

En primer lugar se comienza con el manual del Grabador para una PDA. Una vez que el programa se encuentre instalado en el dispositivo, el usuario se encontrará con las diversas ventanas y opciones que se explican a continuación.

8.1.1 Inicio

Al ejecutar la aplicación Grabador (GBD) se muestra la ventana de comienzo o bienvenida del programa que se muestra a continuación (Imagen 8-1). En ella se pueden observar los diferentes botones con diferentes funciones que se detallan posteriormente en la Imagen 8-2.

Imagen 8-1. Pantalla de bienvenida GBD

Si el usuario ha sido dado de alta anteriormente en el sistema, el nombre de usuario elegido en el momento del registro aparecerá en la lista desplegable (Imagen 8-2.a) y cuando lo haya seleccionado aparecerá como usuario seleccionado (Imagen 8-2.b) mostrando el nombre completo en el cuadro de texto en pantalla. Una vez seleccionado el usuario deseado se puede proceder a realizar el acceso pulsando sobre el botón ***Login***, mediante el cual accederemos a la ventana de usuario (apartado 8.1.3, Imagen 8-8)

Imagen 8-2. Opciones ventana inicio

Un usuario que no se encuentre en la lista desplegable deberá pulsar el botón ***Nuevo Usuario*** (Imagen 8-2.d) para proceder a introducir sus datos en la ventana de registro (apartado 8.1.2, Imagen 8-5.a).

Para eliminar un usuario de la base de datos, el usuario deberá pulsar el ***Menú Herramientas→Borrar Usuario*** (Imagen 8-2.f). El programa pedirá una confirmación al usuario (Imagen 8-3.a) y si éste confirma pulsando el botón ***Sí***, el programa mostrará un mensaje de información indicando que el usuario ha sido eliminado (Imagen 8-3.c) y en caso de que el usuario apriete sobre el botón ***No***, la aplicación mostrará un mensaje (Imagen 8-3.b) informando que el usuario no ha sido eliminado del sistema. En el caso de que el usuario intente borrar un usuario sin haber seleccionado previamente ninguno de la lista de usuarios, el programa mostrará un mensaje (Imagen 8-2.d) indicando al usuario que debe seleccionar uno de los elementos de la lista.

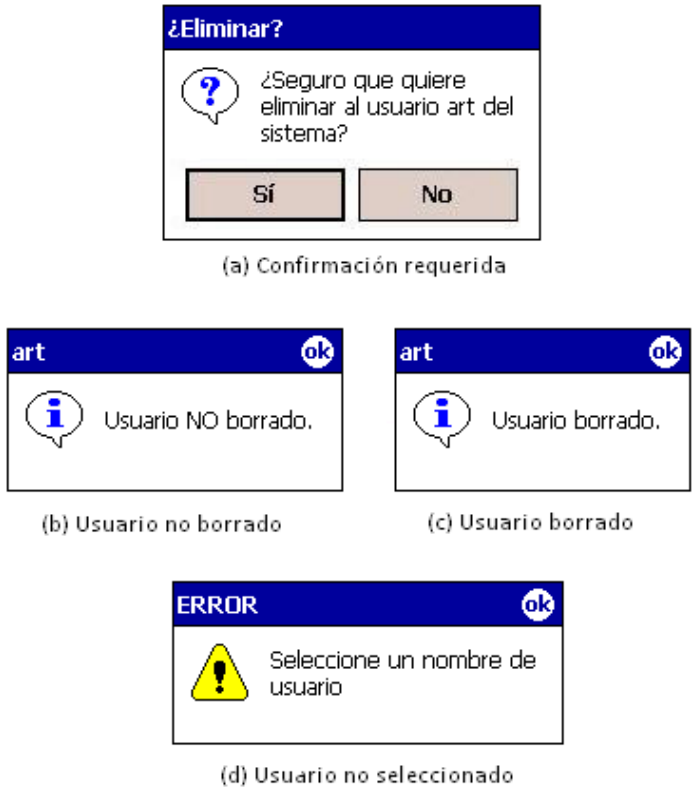


Imagen 8-3. Borrado de usuario

La opción ***Borrar Usuario*** permanece deshabilitada, al igual que el botón de ***Login***, en el caso en que no haya ningún usuario registrado en el sistema o todos hayan sido eliminados del mismo.

En el caso de que el usuario desee conocer cierta información del programa, podrá pulsar la opción del ***Menú Herramientas → Acerca de gbd…*** (Imagen 8-2.g) en donde se le mostrará el nombre del programa, la versión del mismo, el autor, la fecha de creación de la aplicación.

Imagen 8-4. Ventana de información de GBD

Para terminar la ejecución en curso de la aplicación y abandonar el programa Grabador, el usuario deberá pulsar la opción del ***Menú Archivo*** → ***Salir*** **(**Imagen 8-2.e). Una vez seleccionada esta opción, el programa procede a guarda toda la información y finaliza su ejecución.

8.1.2 Nuevo Usuario

Al pulsar el botono ***Nuevo Usuario*** en la ventana de inicio se accede a una nueva ventana (Imagen 8-5a) en la que se le requerirán al usuario diferentes datos para poder completar su registro. Los datos que se piden son su nombre completo, su fecha de nacimiento y su género masculino o femenino. Con el nombre completo que se indica en el cuadro de texto correspondiente, el programa crea automáticamente un nombre de usuario a partir de las iniciales del nombre proporcionado. En caso de creación de un nuevo usuario con las mismas iniciales que uno ya existente, se añadiría al final un guión bajo seguido de un número que iría incrementándose.

En caso de que el usuario no desee registrar un nuevo usuario en este momento, en la ventana mostrada existe la opción de volver a la ventana de inicio anterior simplemente pulsando sobre el botón ***Atrás***.

(a) Registro Nuevo Usuario

(b) Confirmación datos de registro

Imagen 8-5. Ventana Nuevo Usuario

Una vez introducidos los datos necesarios para el registro (Imagen 8-5.a) y pulsando el botón de ***Registrar***, el programa mostrará los datos en una nueva ventana (Imagen 8-5.b) en la que el usuario deberá confirmar o modificar los mismos. Si el usuario pulsa sobre el botón ***Modificar***, la aplicación volverá a mostrar la ventana anterior en la que el usuario podrá variar los datos anteriormente introducidos. En caso de que el usuario esté de acuerdo con los datos mostrados en pantalla deberá pulsar el botón ***Confirmar***. Al confirmar el nuevo registro el programa mostrará un mensaje informando de este hecho (Imagen 8-6).

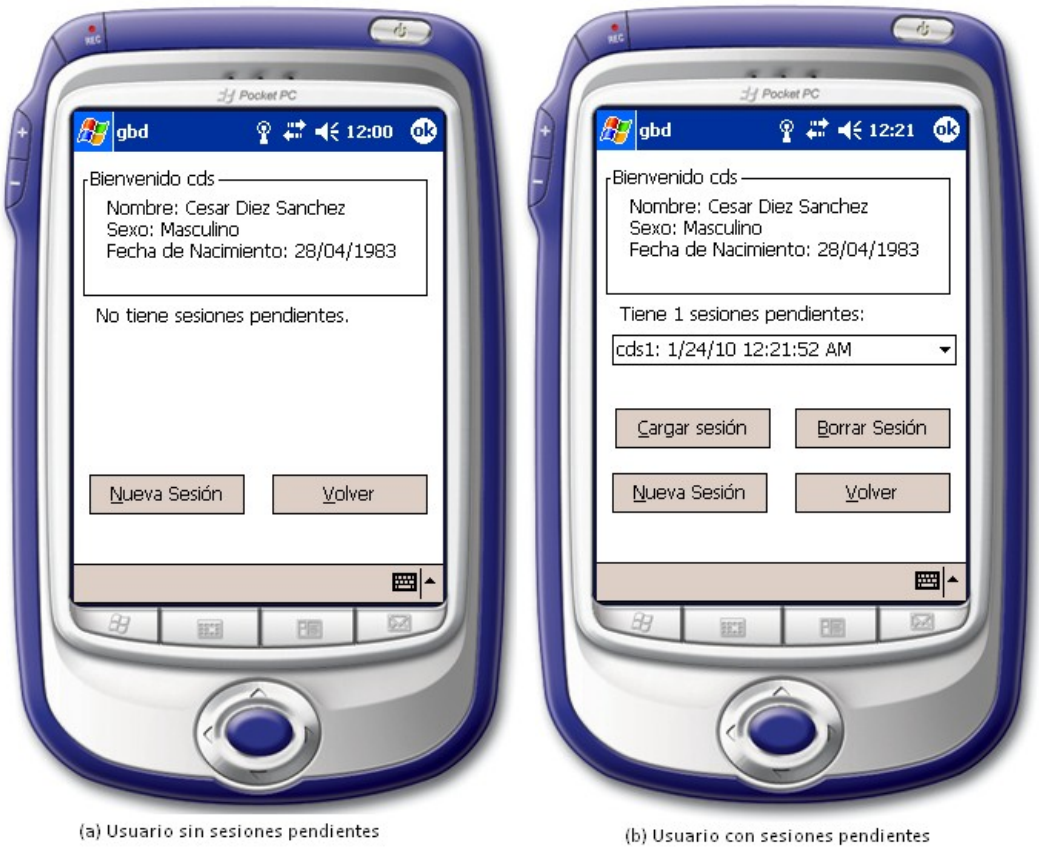
Imagen 8-6. Nuevo usuario creado

En caso de que el usuario haya intentado registrarse sin haber completado todos los parámetros pedidos, el programa mostrara un aviso (Imagen 8-7) al usuario indicándole que debe rellenar todos los campos para poder completar el registro. En este momento el usuario podrá ***Reintentar*** el registro o ***Cancelar*** el mismo de modo que volvería la ventana de bienvenida.

Imagen 8-7. Datos de registro incompletos

8.1.3 Ventana de Usuario

En esta ventana aparecen cargados todos los datos del usuario que acaba de acceder al sistema, ya sea desde un nuevo registro o desde la pantalla de acceso mediante un acceso convencional mediante el botón ***Login*** en la ventana inicial. En esta pantalla, según se puede ver en la Imagen 8-8, se cargan los datos del usuario y se informa del número de sesiones pendientes que tiene. En caso de no tener ninguna sesión pendiente, sólo aparecen habilitados los botones de ***Nueva Sesión*** y ***Volver***. El primero se ha de pulsar cuando se quiera iniciar una nueva sesión de grabaciones (ver apartado 8.1.4), y el segundo para terminar y volver a la pantalla inicial del programa.



(a) Usuario sin sesiones pendientes

(b) Usuario con sesiones pendientes

Imagen 8-8. Ventana de Usuario

En caso de existir sesiones pendientes por parte del usuario conectado, éstas aparecerán el desplegable de la Imagen 8-8.b, así como aparecerán habilitados los botones ***Cargar Sesión*** y ***Borrar Sesión*****.** Con el primer botón se pasará a la pantalla de grabación para continuar la sesión que ha quedado pendiente (ver apartado 8.1.5) mientras que si el usuario pulsa sobre el botón de ***Borrar Sesión***, automáticamente el programa eliminará dicha sesión pero dejando las grabaciones realizadas en el disco de la PDA. En caso de que el usuario desee borrar una de las sesiones pendientes, se pedirá confirmación al usuario de igual manera que la explicada para el caso del borrado de usuario (Imagen 8-9).

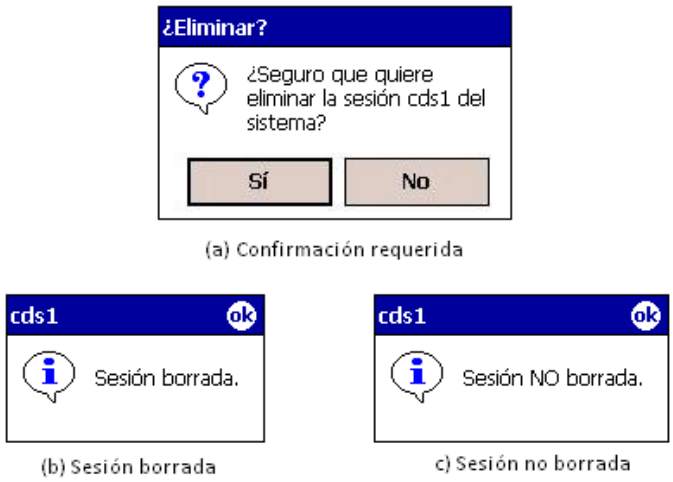


Imagen 8-9. Borrado de sesión

8.1.4 Nueva Sesión

Cuando el usuario desea comenzar la grabación de una nueva sesión se encontrará en la pantalla que se muestra a continuación:

Imagen 8-10. Ventana Nueva Sesión

Al principio de la pantalla aparece un campo no editable que muestra el nombre de la futura sesión. Este nombre se codifica de forma automática y no se puede modificar. Está formado por el nombre del usuario que la crea seguido de X, siendo X el número de sesión histórica del usuario.

Como segunda opción se pide que se indique el tipo de micrófono que se va a usar en esta sesión. Esta decisión es importante pues en función de ésta se codificará el nombre de los ficheros de audio generados y esta información se utilizará más adelante en el tratamiento de

la base de datos. La opción ***PDA*** hace referencia al uso del micrófono integrado del dispositivo. Se deberá seleccionar esta opción si no dispone de ningún micrófono alternativo. La opción ***BlueTooth*** se utilizará cuando se cuente con un micrófono o manos libres conectado a la PDA. En cuanto a la configuración, se podrá seleccionar la ***Normal*** o la ***Avanzada***.

Una vez seleccionadas todas las opciones se pulsa el botón ***Crear*** y se informa del registro de la nueva sesión si se ha llevado a cabo de forma satisfactoria. Si el usuario ha pulsado el botón antes de haber completado todo el formulario, se le mostrará un mensaje (Imagen 8-11) indicándole que faltan datos por rellenar. De esta manera, el usuario podrá volver a introducir los datos que falten o cancelar la creación de una nueva sesión y volver a la ventana anterior.

Imagen 8-11. Formulario nueva sesión incompleto

En caso de que el usuario haya elegido la configuración normal, ver apartado 8.1.4.1, pasa a la ventana de sesión cargada, apartado 8.1.5, y en el caso de configuración avanzada, apartado 8.1.4.2, se pasa primero por la ventana de sesión avanzada, Imagen 8-12.

8.1.4.1 Configuración normal

Cuando se crea una nueva sesión con la configuración normal, se genera la estructura de directorios nueva para almacenar las pistas que se graben en la sesión. Para ello se crea en la carpeta *data* del programa una subcarpeta con el nombre de usuario y dentro de ella otra con el nombre de la sesión. Dentro de esta carpeta se almacena un fichero de configuración con el nombre de la sesión y extensión *.cfg* que contiene la información relativa a la misma (Tabla 8-1) basada en el fichero de configuración por defecto sito en la carpeta *bin* del programa, *default.cfg* (ver Tabla 8-2).

cds1.cfg	
Fichero de frases:	\Archivos de programa\gbd\bin\sentences\Diccio.snt
Frecuencia de muestreo:	8000
Directorio Almacenamiento:	\Archivos de programa\gbd\data

Tabla 8-1. Fichero de configuración normal

En el fichero aparecen las rutas absolutas (en *default.cfg* aparecen relativas) del fichero de frases que se utilizará en la sesión y del directorio raíz de almacenamiento, donde se creará la estructura de directorios descrita. Además aparece la frecuencia de grabación de los archivos de audio. En esta versión del programa no es posible modificar esta frecuencia de grabación.

default.cfg	
Fichero de frases:	bin\sentences\Diccio.snt
Frecuencia de muestreo:	8000
Directorio Almacenamiento:	data

Tabla 8-2. Fichero de configuración por defecto

8.1.4.2 Configuración avanzada

Cuando se selecciona la opción de configuración avanzada, se lanza en el programa una nueva ventana, Imagen 8-12. En ella se ofrece una forma sencilla de modificar el fichero de configuración del usuario sin necesidad de utilizar un editor. Hacer notar que en este caso sólo se modifica el fichero de configuración de la sesión a crear (*session.cfg*), y no el fichero de configuración por defecto (*default.cfg*).

Imagen 8-12. Ventana configuración avanzada

Para modificar las rutas del fichero de frases y del directorio de almacenamiento se ofrecen sendas ventanas, lanzadas al presionar el botón ***Examinar*** cercano al campo de texto correspondiente. Se pueden ver ambas en la Imagen 8-13.

En la subfigura primera se muestra la ventana de selección del fichero de frases. La pantalla aparece inicialmente dividida en dos, de forma que en la superior se puede navegar cómodamente por los directorios y en la inferior se ve el contenido de los mismos para seleccionar el fichero de frases deseado. En la barra de herramientas inferior, aparecen una serie de botones que se listan a continuación.

	Cuando ya está seleccionado un fichero de frases, se pulsa para confirmar.
	Utilizado para volver a la pantalla anterior sin seleccionar ningún fichero.
	Vista por defecto. Aparece la pantalla dividida, carpetas en la parte superior y archivos en la inferior.
	Vista por carpetas. La pantalla aparece dedicada exclusivamente a la navegación por carpetas.
	Listado de archivos. La pantalla muestra únicamente el listado de archivos.

Tabla 8-3. Barra herramientas GBD

(a) Examinar fichero de frases

(b) Examinar directorio de almacenamiento

Imagen 8-13. Selección de ficheros y directorios

En la segunda subfigura de la Imagen 8-13, los botones de división de pantalla y de listado están desactivados de modo que sólo se puede ver el árbol de directorios, pues el objetivo es escoger una carpeta para alojar la estructura.

Aparece además en la pantalla de configuración avanzada un cuadro de selección para la frecuencia de grabación. En esta versión del programa sólo se permite una frecuencia.

Para terminar se pulsa el botón ***Aceptar*** de la pantalla y se dará paso a la pantalla de sesión cargada, apartado 8.1.5. En caso de que el usuario desee volver a la ventana anterior, deberá pulsar el botón ***Atrás***.

8.1.5 Sesión cargada

Esta ventana mostrada en la Imagen 8-14, la aplicación carga los datos de la sesión actual y tiene como objeto ayudar a identificar al usuario la sesión, así como de recordar con qué micrófono debe grabar y dónde puede encontrar los audios generados.

En el cuadro de texto se muestra, como se puede ver, la fecha de creación de la sesión, la última frase que se ha grabado y a partir de la cual se iniciarán las grabaciones, el tipo de micrófono que se debe utilizar y la ruta completa del fichero de configuración, que es además la ruta del directorio base para el almacenamiento de los archivos de audio.

Esta sencilla ventana tiene sólo dos botones: ***Continuar grabaciones*** y ***Volver***. El primero da paso a la grabación, apartado 8.1.6, y el segundo vuelve a la pantalla de bienvenida o de usuario, Imagen 8-8.

Imagen 8-14. Sesión cargada

8.1.6 Grabación

Al llegar a esta ventana, mostrada en la Imagen 8-15, el usuario se encuentra en disposición de grabar las secuencias del fichero de frases escogido para la sesión actual.

Imagen 8-15. Pantallas de grabación

La primera ventana que aparece es la de la Imagen 8-15.a que muestra la información sobre los cuadros de texto y su funcionalidad. Así en el primer cuadro de texto de los tres existentes (todos ellos no editables) se muestra el número de la última frase grabada. Ese número es equivalente a la línea del fichero de texto donde se almacenan las frases y aparecerá un 0 si no se ha almacenado ninguna frase en la sesión. En el segundo cuadro aparecerá el texto de la frase o palabra que se deberá leer y por tanto grabar y en el último se muestra el nombre del fichero que almacenará la pista de audio generada.

Al pulsar ***Comenzar*** se carga la primera frase, o la siguiente a la última grabada si es una sesión pendiente. Aparece entonces en el primer cuadro el número de frase, en el segundo el texto de la misma y en el tercero el nombre del archivo de audio. Además, se cambia el botón Comenzar por el de ***Siguiente Frase*** tal y como se ve en la Imagen 8-15.b. Para comenzar la grabación, el usuario debe pulsar sobre el botón ***Siguiente Frase*** y en ese instante el programa indicará el comienzo de la grabación con un pitido. El locutor dispondrá de un periodo de 3 segundos para pronunciar la frase o palabra a grabar y al final de este tiempo, el programa volverá a emitir un pitido indicando el fin de la grabación a la vez que mostrará en pantalla la frase siguiente a ser grabada. Cuando el usuario ha terminado de grabar todas las palabras o frases de una sesión se le informa (Imagen 8-16) de que la sesión ha sido grabada con éxito y el programa sigue su curso en la ventana de bienvenida.

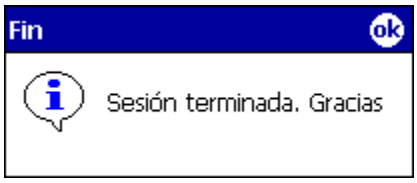


Imagen 8-16. Fin grabaciones

Para salir de las grabaciones, no hay más que pulsar el botón de ***Salir***, y se regresará a la pantalla de usuario. En el caso de pulsar este botón en mitad de una sesión, ésta quedara pendiente en la frase en que se abandonó y podrá ser retomada su grabación en cualquier otro instante.

8.2 Parametrizador

Para continuar con los manuales de usuario que se muestran en este capítulo pasamos a detallar todas las funcionalidades del programa Parametrizador. Como ya se ha explicado a lo largo de esta memoria, este programa es que el crea los archivos de parámetros MFCC a partir de los ficheros de voz generados por el usuario, normalmente a través del anterior programa GBD.

8.2.1 Inicio

Al ejecutar la aplicación, el programa mostrará la ventana de inicio (Imagen 8-17) en la que el usuario deberá elegir los ficheros *lista archivos voz* y *lista archivos de parámetros*.

El archivo *lista archivos voz* contiene todos los archivos de voz que se pretende que sean parametrizados con la ruta absoluta de su ubicación en la PDA, tal y como se muestra en la Tabla 8-4.

De la misma manera, el usuario debe seleccionar la *lista archivos de parámetros* en donde aparecen todas las rutas absolutas para cada uno de los ficheros de voz anteriormente mencionados en donde se almacenarán los diferentes archivos de parámetros. Se puede ver un ejemplo de este archivo en la Tabla 8-5.

Imagen 8-17. Ventana de inicio Parametrizador

Según la tabla siguiente, los ficheros de voz que se parametrizarían serian las 3 primeras frases grabadas en la primera sesión de grabación del usuario *cds*:

listaVozEjemplo.txt
\Storage Card\BBDD\cds1_1
\Storage Card\BBDD\cds1_2
\Storage Card\BBDD\cds1_3

Tabla 8-4. Lista Voz Parametrizador

listaParamEjemplo.txt
\Storage Card\Parametros\cds1_1.mfcc
\Storage Card\Parametros\cds1_2.mfcc
\Storage Card\Parametros\cds1_3.mfcc

Tabla 8-5. Lista Parametros Parametrizador

En la tabla anterior se muestra un ejemplo de lista en la que cada archivo de voz parametrizado aparece con la ruta absoluta en la que quedará almacenado.

Ambos ficheros de texto deben ser creados por el usuario utilizando las rutas absolutas.

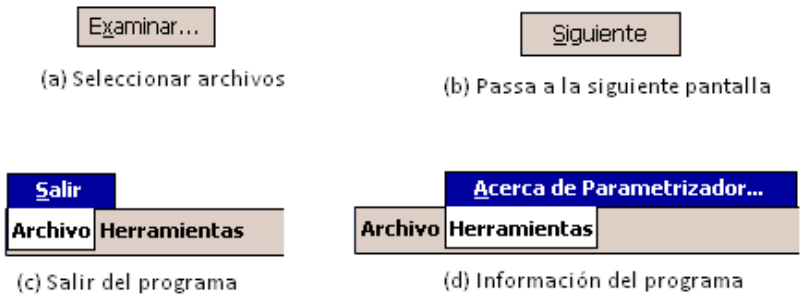


Imagen 8-18. Opciones ventana inicio Parametrizador

En la ventana de inicio del Parametrizador, el usuario debe seleccionar los archivos donde aparecen las listas de archivos de voz y de parámetros que quiere utilizar en cada uso. Al pulsar sobre el botón ***Examinar*** (Imagen 8-18.a) se mostrara la ventana del explorador que aparece en la Imagen 8-19.

Imagen 8-19. Examinar archivos Parametrizador

La pantalla aparece inicialmente dividida en dos, de forma que en la superior se puede navegar cómodamente por los directorios y en la inferior se ve el contenido de los mismos para seleccionar el fichero de frases deseado. En la barra de herramientas inferior, aparecen una serie de botones que se listan a continuación.

ok	Cuando ya está seleccionado un fichero, se pulsa para confirmar.
	Utilizado para volver a la pantalla anterior sin seleccionar ningún fichero.
	Vista por defecto. Aparece la pantalla dividida, carpetas en la parte superior y archivos en la inferior.
	Vista por carpetas. La pantalla aparece dedicada exclusivamente a la navegación por carpetas.
	Listado de archivos. La pantalla muestra únicamente el listado de archivos.

Tabla 8-6. Barra herramientas Parametrizador

Una vez que el usuario ha seleccionado ambas listas de ficheros, el botón ***Siguiente*** (Imagen 8-18.b) pasará a poder ser pulsado y el usuario pasara a la ventana de configuración (ver apartado 8.2.2).

En el caso de que el usuario desee conocer cierta información del programa, podrá pulsar la opción del ***Menú Herramientas → Acerca de Parametrizador…*** (Imagen 8-18.d) en donde se

le mostrará el nombre del programa, la versión del mismo, el autor, la fecha de creación de la aplicación.

Imagen 8-20. Acerca de Parametrizador

Para terminar la ejecución en curso de la aplicación y abandonar el programa Grabador, el usuario deberá pulsar la opción del ***Menú Archivo*** → ***Salir*** (Imagen 8-18.c). Una vez seleccionada esta opción, el programa procede a guarda toda la información y sale del programa.

8.2.2 Configuración

En la ventana mostrada en la Imagen 8-21 se muestran las opciones de Parametrización que el usuario puede elegir para realizar la misma en los ficheros de voz seleccionados. En esta ventana, el usuario podrá elegir entre realizar la parametrización en Punto Flotante o en Punto Fijo.

Imagen 8-21. Ventana de configuración Parametrizador

Pulsando el botón ***Atrás***, la ejecución volverá a la pantalla de inicio mostrada anteriormente mientras que, una vez que el usuario haya elegido el tipo de parametrización, el botón ***Comenzar*** pasará a mostrarse habilitado de modo que la parametrización podrá tener lugar.

8.2.3 Parametrización

Cuando el usuario pulsa el botón ***Comenzar*** en la ventana mostrada en la Imagen 8-21, el programa pedirá confirmación para que la parametrización empiece. Para ello, el usuario deberá pulsar el botón ***ok*** del mensaje mostrado en la Imagen 8-22.

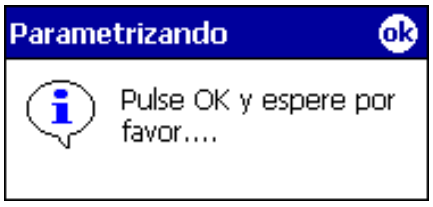


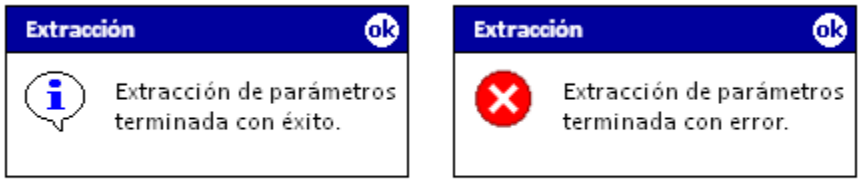
Imagen 8-22. Comenzar Parametrización

(a) Parametrizando en Punto Flotante (b) Parametrizando en Punto Fijo

Imagen 8-23. Parametrizando

En la Imagen 8-23.a podemos ver el mensaje que se muestra por pantalla mientras la parametrización va teniendo lugar en el caso de haber elegido una parametrización en Punto Flotante. En el caso de que el usuario hubiera escogido realizar la parametrización en Punto Fijo, el mensaje variaría ligeramente, tal y como se muestra en la Imagen 8-23.b.

Una vez la parametrización concluye, el programa mostrara uno de los dos mensajes mostrados en la Imagen 8-24 dependiendo de si la parametrización ha finalizado de forma exitosa o si por el contrario ha ocurrido algún error durante el proceso.



(a) Final exitoso (b) Final con error

Imagen 8-24. Final de la parametrización

8.3 Adaptador

En este punto se recuerda al lector que el Adaptador que se va a presentar a continuación es el programa que realiza la adaptación del modelo general de mundo (UBM) de manera que quede adaptado a las características propias de cada locutor. Para el correcto funcionamiento del programa será necesario, por tanto, indicar al programa el modelo de mundo que debe utilizar para la adaptación así como los ficheros de parámetros del locutor del que se desea obtener su modelo de voz.

8.3.1 Inicio

Al abrir el programa, éste mostrara la ventana de inicio que se presenta en la Imagen 8-25, en la que el usuario debe seleccionar el fichero de configuración escogido para la sesión de adaptación que desea que el programa realice.

Imagen 8-25. Ventana de inicio Adaptador

El fichero de configuración debe estar creado previamente por el usuario siguiendo la estructura del ejemplo mostrado en la Tabla 8-7 para que el programa lo interprete correctamente.

En la Imagen 8-26 se pueden ver las diferentes opciones de que dispone el usuario en esta ventana de inicio. Al comenzar el programa por primera vez y cada vez que no se encuentre seleccionado ningún fichero de configuración, el botón ***Comenzar adaptación*** (Imagen 8-26.b) aparecerá deshabilitado de manera que únicamente cuando el usuario elija el fichero de configuración deseado, éste botón aparecerá habilitado.

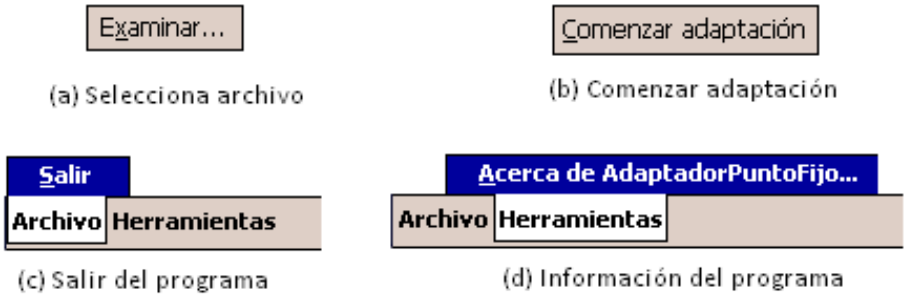


Imagen 8-26. Opciones ventana inicio Adaptador

Para seleccionar el fichero de configuración, el usuario deberá pulsar el botón ***Examinar*** (Imagen 8-26.a) y se abrirá la ventana habitual del explorador de ficheros mostrado en la Imagen 8-27.

Imagen 8-27. Seleccionar archivo configuración Adaptador

En la ventana de selección de fichero, el usuario navegará por las diferentes carpetas que se muestran en forma de árbol en la parte superior de la imagen. En la parte inferior de la imagen se mostrarán los ficheros situados dentro de la carpeta que se encuentre seleccionada en ese momento. De esta manera el usuario seleccionará el fichero deseado de una forma muy intuitiva.

Las diferentes opciones de la barra de herramientas presente en la parte inferior de la imagen, se detallan a continuación:

Cuando ya está seleccionado un fichero, se pulsa para confirmar.


Utilizado para volver a la pantalla anterior sin seleccionar ningún fichero.


Vista por defecto. Aparece la pantalla dividida, carpetas en la parte superior y archivos en la inferior.


Vista por carpetas. La pantalla aparece dedicada exclusivamente a la navegación por carpetas.


Listado de archivos. La pantalla muestra únicamente el listado de archivos.

Imagen 8-28. Barra de herramientas Adaptador

El usuario podrá comprobar la versión, el autor o la fecha de creación del producto que está instalado en su PDA de una manera sencilla pulsando la opción ***Acerca de AdaptadorPuntoFijo…*** del menú ***Herramientas*** (Imagen 8-26.d). La ventana que le aparecerá al usuario será la que se muestra en la Imagen 8-29.

Imagen 8-29. Acerca de Adaptador

Para abandonar la ejecución del programa en el momento que el usuario desee, éste no tendrá más que pulsar sobre la opción ***Salir*** del menú ***Archivo*** (Imagen 8-26.c).

8.3.2 Adaptación

Para que el proceso de adaptación tenga lugar, el usuario deberá seleccionar un fichero de configuración que tenga una estructura como la que se muestra a continuación:

config.txt
#Directorio de almacenamiento de los modelos de mundo.
DIR_MUNDO \Archivos de programa\Verificador\bin\Modelos\
#Directorio para los modelos de salida adaptados.
DIR_SALIDA \Archivos de programa\Adaptador\Salida\
#Directorio de las listas de ficheros de parámetros de voz de usuarios.
DIR_LISTA \Archivos de programa\Adaptador\bin\Listas\
#Nombre del fichero de mundo sito en el directorio indicado antes.
FILE_MUNDO world
#Nombre del fichero de salida para el modelo adaptado al usuario.
FILE_SALIDA cds
#Nombre del fichero que contiene la lista de los ficheros de usuario.
FILE_LISTA lista.txt

Tabla 8-7. Ejemplo fichero de configuración Adaptador

En el ejemplo mostrado se deben introducir las variables (DIR_MUNDO, DIR_SALIDA, DIR_LISTA, FILE_MUNDO, FILE_SALIDA y FILE_LISTA) tal y como aparecen en el ejemplo si variar una letra, ya que cualquier cambio provocará que el programa no interprete correctamente el fichero y de un error en la ejecución. Asimismo, las rutas de los directorios deben ser rutas absolutas y no relativas para que el programa se ejecute correctamente. En lo relativo al nombre de los ficheros y a su ubicación en la PDA, el usuario podrá situarlo donde más desee sin que tenga que elegir obligatoriamente la que se muestra en el fichero de ejemplo.

Como ya se ha explicado anteriormente, para el correcto funcionamiento de la aplicación, el usuario deberá disponer en la PDA del modelo de mundo UBM correctamente generado en Punto Fijo. Este será el modelo que el programa adapte al usuario según los ficheros de parámetros indicados en la lista de ficheros del usuario que se desee adaptar. Un ejemplo de este archivo de lista se muestra en la siguiente tabla:

lista.txt
\Storage Card\Parametros\cds1_1.mfcc
\Storage Card\Parametros\cds1_2.mfcc
\Storage Card\Parametros\cds1_3.mfcc

Tabla 8-8. Ejemplo fichero lista

El fichero *lista.txt* es un ejemplo, el usuario podrá crear su propio archivo en el que deberá incluir todos los ficheros de parámetros que desee utilizar para la adaptación (se recomienda poner todos los ficheros de que se disponga) con sus rutas absolutas.

Ambos ficheros, el de configuración y la lista, deberán tener extensión TXT para que el programa funcione debidamente.

Una vez que el usuario haya seleccionado el fichero de configuración con todos los parámetros necesarios, deberá proceder a pulsar el botón ***Comenzar adaptación*** para que el programa pueda leer todos los parámetros de entrada y proceder a la adaptación, sin embargo, antes de comenzar el proceso le pedirá confirmación al usuario (Imagen 8-30) que deberá pulsar el botón ***OK*** para continuar con la adaptación o ***Cancelar*** para volver a la ventana inicial.

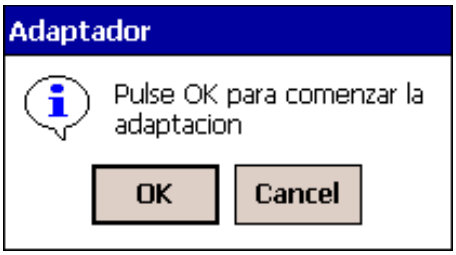


Imagen 8-30. Confirmación Adaptación

Una vez que la adaptación ha terminado, el programa mostrara un mensaje de información al usuario indicando que la adaptación ha terminado de forma exitosa, tal y como se muestra en la Imagen 8-31.

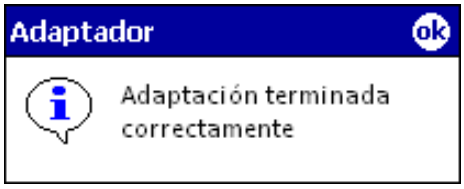


Imagen 8-31. Adaptación terminada

8.4 Verificador

Esta es la aplicación principal y para la cual se ha desarrollado todo el presente Proyecto. Se trata de un programa que se encargará de determinar si el usuario que intenta conectarse por medio de un nombre de usuario, es realmente quien dice ser. Para ello, le pedirá al usuario que realice una breve prueba de voz de modo que pueda comparar con el modelo de usuario elegido. Si la comparación sale favorable, el usuario sería aceptado dentro del sistema mientras que si por el contrario, la comparación sale desfavorable, el usuario no lograría acceder al sistema.

8.4.1 Inicio

Al ejecutar la aplicación Verificador de Locutor se muestra la ventana de comienzo o bienvenida del programa que se presenta a continuación (Imagen 8-32). En ella se pueden observar los diferentes botones con diferentes funciones que se detallan posteriormente en la Imagen 8-33.

Imagen 8-32. Ventana de inicio Verificador

Si el usuario ha sido dado de alta anteriormente en el sistema, el nombre de usuario elegido en el momento del registro aparecerá en la lista desplegable (Imagen 8-33.a) y cuando lo haya seleccionado aparecerá como usuario seleccionado (Imagen 8-33.b) mostrando el nombre completo en el cuadro de texto en pantalla. Una vez seleccionado el usuario deseado se puede proceder a realizar el acceso pulsando sobre el botón ***Login***, mediante el cual accederemos a la ventana de verificación (apartado 8.4.3 Imagen 8-41).

Por defecto, el programa lee un fichero de configuración que debe crear o modificar el usuario si así lo desea, en el que se indican varios parámetros listados a continuación:

- Modelo de Mundo. Indispensable para el proceso de verificación.
- Ruta modelo de locutor. El programa buscará en esta ruta el modelo de locutor.
- Ficheros auxiliares (FILE_BEEP, FILE_CBIN, FILE_DCT, FILE_MEL, FILE_HAM).
- Tabla de Viterbi utilizada.
- Fichero de parámetros. En este fichero se guardará la parametrización de la muestra del locutor cuando intenta verificarse.
- Fichero de voz. El programa guarda también la captura de voz realizada al usuario.

Un ejemplo de dicho fichero de configuración *config_file.txt* se muestra en la tabla a continuación:

config_file.txt
#Directorio de almacenamiento de los modelos de mundo. DIR_MUNDO \Archivos de programa\Verificador\bin\Modelos\ *#Directorio de almacenamiento de los modelos de locutor.* DIR_USUARIO \Archivos de programa\Verificador\bin\Modelos\ *#Directorio para el fichero de parámetros generado.* DIR_VOZPAR \Archivos de programa\Verificador\data\ *#Directorio para los ficheros auxiliares.* DIR_ENTRADA \Archivos de programa\Verificador\bin\ *#Directorio para los modelos de salida adaptados.* DIR_SALIDA \Archivos de programa\Adaptador\Salida\ *#Directorio de almacenamiento de la TablaViterbi utilizada.* DIR_TABLA \Archivos de programa\Verificador\bin\Tablas\ *#Nombre del fichero de mundo sito en el directorio indicado antes.* FILE_MUNDO world *#Nombre del fichero de parámetros elegido.* FILE_VOZPAR graba.mfcc *#Nombre del fichero de salida.* FILE_SALIDA salida.txt *#Nombre del fichero que contiene la tabla de Viterbi.* FILE_TABLA tablaQ7.txt *#Nombre del fichero que contiene la grabación de voz del usuario.* FILE_VOICE graba.voz *#Nombre del fichero que contiene el pitido de inicio/fin de grabación.* FILE_BEEP pitido.wav *#Nombre del primer fichero auxiliar.* FILE_CBIN Cbin_23.txt *#Nombre del segundo fichero auxiliar.* FILE_DCT DCT_LUT.txt *#Nombre del tercer fichero auxiliar.* FILE_MEL Fix_MelFbank_23.txt *#Nombre del cuatro fichero auxiliar.* FILE_HAM HammingWindow.txt

Tabla 8-9. Ejemplo fichero de configuración Verificador

El usuario podrá modificar este fichero, pero a no ser que el usuario sea un experto en la materia, se recomienda no modificar los parámetros proporcionados en este ejemplo.

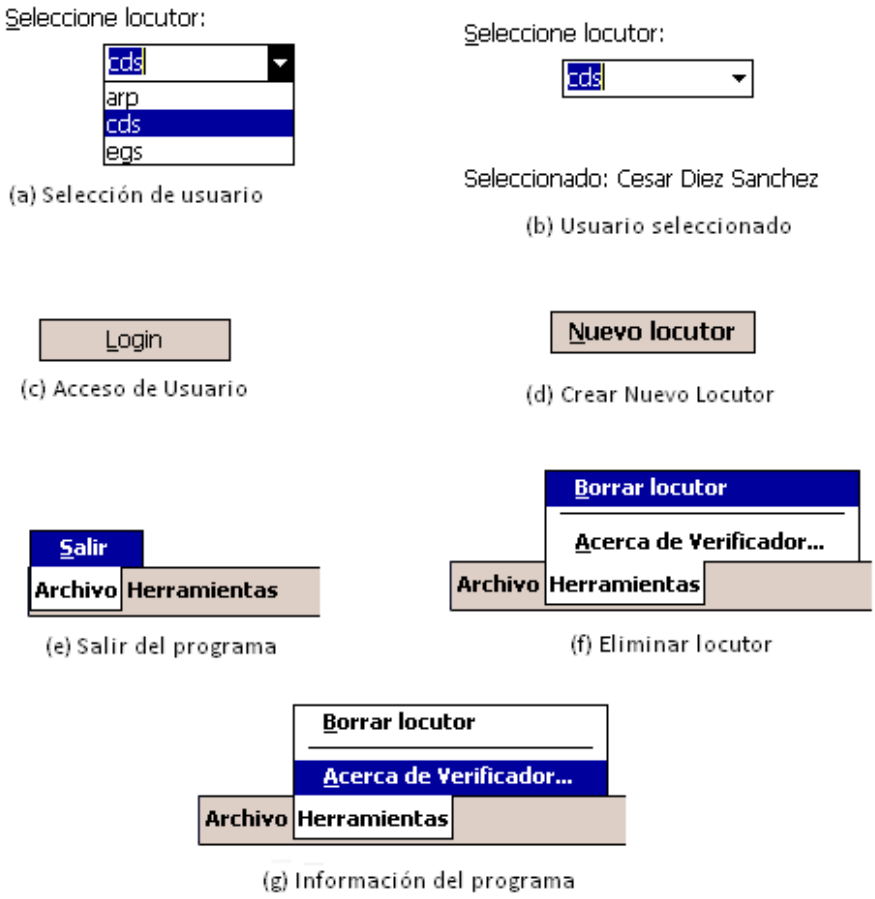


Imagen 8-33. Opciones ventana inicio Verificador

Un usuario que no se encuentre en la lista desplegable deberá pulsar el botón ***Nuevo Locutor*** (Imagen 8-33.d) para proceder a introducir sus datos en la ventana de registro (apartado 0, Imagen 8-38.a).

Para eliminar un locutor de la base de datos, el usuario deberá pulsar el ***Menú Herramientos→Borrar Locutor*** (Imagen 8-33.f). El programa pedirá una confirmación al usuario (Imagen 8-34.a) y si éste confirma pulsando el botón ***Sí***, el programa mostrará un mensaje de información indicando que el usuario ha sido eliminado (Imagen 8-34.b) y en caso de que el usuario apriete sobre el botón ***No***, la aplicación mostrará un mensaje (Imagen 8-34.c) informando que el usuario no ha sido eliminado del sistema.

La opción ***Borrar Locutor*** permanece deshabilitada, al igual que el botón de ***Login***, en el caso en que no haya ningún usuario registrado en el sistema o todos hayan sido eliminados del mismo.

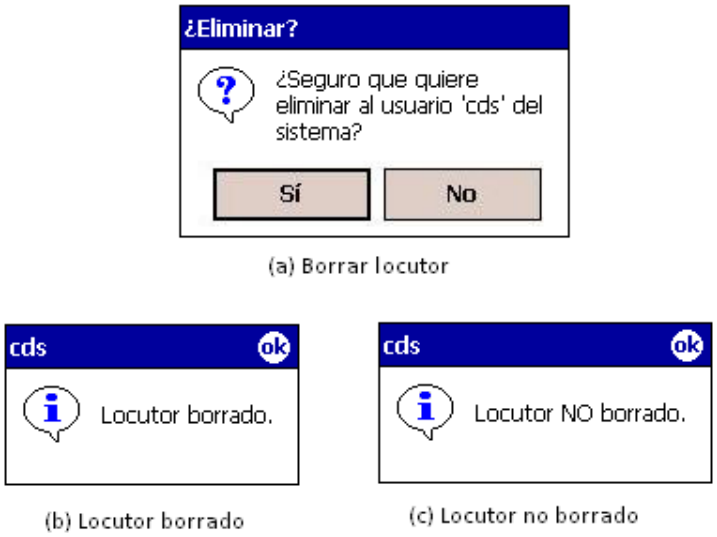


Imagen 8-34. Borrado de locutor

En el caso de que el usuario desee conocer cierta información del programa, podrá pulsar la opción del ***Menú Herramientas → Acerca de Verificador…*** (Imagen 8-33.g) en donde se le mostrará el nombre del programa, la versión del mismo, el autor, la fecha de creación de la aplicación.

Imagen 8-35. Acerca de Verificador

Para terminar la ejecución en curso de la aplicación y abandonar el programa Grabador, el usuario deberá pulsar la opción del ***Menú Archivo → Salir*** (Imagen 8-33.e). Una vez

seleccionada esta opción, el programa le preguntará al usuario (únicamente en caso de haber añadido o eliminado algún nuevo locutor) si desea guardar dichas variaciones en el archivo de locutores *speaker.spk* (Imagen 8-36).

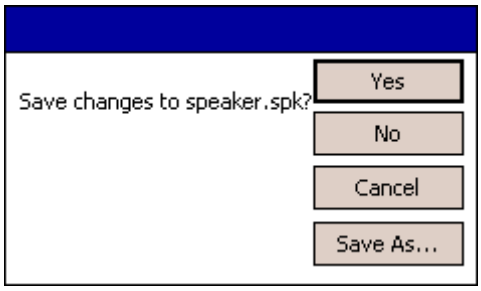


Imagen 8-36. Salir de Verificador

En esta ventana, el usuario podrá guardar los cambios en el archivo indicado en el mensaje (pulsando ***Si***), descartar dichos cambios manteniendo el fichero anterior (pulsando ***No***), cancelar la salida del programa (pulsando ***Cancelar***) o guardar los cambios producidos en otro fichero mediante la opción ***Guardar Como***.

Si el usuario decide renombrar el fichero de locutores con otro nombre, el programa le mostrará la ventana siguiente, en la que el usuario podrá elegir el nuevo nombre del fichero de locutores, así como la carpeta en la que lo desea almacenar.

Imagen 8-37. Guardar fichero como

Esta opción, a pesar de estar disponible, no tiene efecto en la versión realizada del Verificador ya que actualmente, el fichero que contiene los locutores no se puede modificar de nombre. De esta manera, el programa utilizará siempre el denominado *speaker.spk* (ver Tabla 8-10) que debe encontrarse en la carpeta *bin* del directorio donde se encuentre instalado el Verificador.

speaker.spk
Usuario:{Usuario, Nombre, Fecha nacimiento[dd/mm/yyyy], Sexo, Umbral} agm:{agm, Alberto Garcia Moreno, [02/01/1900], Masculino, 56} cds:{cds, Cesar Diez Sanchez, [28/04/1983], Masculino, 99} erp:{erp, Elena Romero Perales, [24/07/1983], Femenino, 62}

Tabla 8-10. Ejemplo fichero de locutores

8.4.2 Nuevo locutor

Una vez el modelo de locutor ha sido creado en el proceso de adaptación, es el usuario el que debe crear su perfil de usuario dentro de este programa Verificador. Para ello deberá acceder a la ventana de registro de locutor (Imagen 8-38.a) mediante la opción ***Nuevo Locutor*** en la ventana de inicio de la aplicación.

(a) Registro nuevo locutor (b) Confirmación registro

Imagen 8-38. Ventana Nuevo Locutor

En esta ventana, el usuario deberá introducir todos sus datos de manera que puedan quedar registrados en el fichero *speaker.spk* mostrado anteriormente (ver Tabla 8-10). En la presente versión del programa, el usuario debe introducir entre estos parámetros, el umbral de decisión que el programa utilizará para determinar si el usuario será finalmente verificado o no. Este parámetro se le proporcionaría a cada nuevo usuario ya que para ello se deben realizar diferentes pruebas para determinarlo (ver apartado 3.1.3.3).

Cuando el usuario ha introducido todos sus datos procederá a pulsar el botón de ***Registrar*** para pasar a la siguiente ventana (Imagen 8-38.b) en la que deberá ***Confirmar*** los datos o ***Modificar*** los mismos si así lo desea. Al confirmar el nuevo registro, el programa mostrará un mensaje informando de este hecho (Imagen 8-39).

Imagen 8-39. Nuevo registro creado

En caso de que el usuario haya intentado registrarse sin haber completado todos los parámetros pedidos, el programa mostrara un aviso (Imagen 8-40) al usuario indicándole que debe rellenar todos los campos para poder completar el registro. En este momento el usuario podrá ***Reintentar*** el registro o ***Cancelar*** el mismo de modo que volvería la ventana de bienvenida.

Imagen 8-40. Datos de registro incompletos

Una vez el registro ha sido efectuado, el programa volverá a mostrar la ventana de inicio del programa con la novedad de que en la lista desplegable se podrá elegir al nuevo locutor creado. En caso de no haber ningún otro locutor anteriormente, el botón de ***Login*** habrá pasado a mostrarse habilitado de manera que el usuario podrá intentar verificarse con el nuevo locutor creado.

8.4.3 Verificación

Para acceder a la pantalla de verificación (Imagen 8-41), el usuario deberá seleccionar el modelo de locutor con el que desea verificarse. Para ello deberá seleccionarlo de la lista desplegable y proceder a continuación a pulsar sobre el botón ***Login***.

Desde esta vista, el usuario podrá volver hacia la ventana anterior mediante el botón ***Atrás*** sin que haya tenido lugar el proceso de verificación.

Imagen 8-41. Ventana de verificación

Sin embargo, si el usuario desea acceder al sistema deberá prestar atención a las indicaciones mostradas en el mensaje de esta ventana. En él se le indica que para comenzar el proceso de verificación deberá pulsar el botón ***Comenzar*** y esperar al pitido que indique el inicio de la grabación. Una vez el usuario haya escuchado el pitido deberá comenzar a hablar durante un tiempo de 3 segundos. Al final de los mismos volverá a sonar otro pitido indicando el final de la grabación y se mostrará el resultado obtenido de la verificación. El usuario podrá comprobar si ha sido admitido dentro del sistema (Imagen 8-43) o si por el contrario ha sido rechazado (Imagen 8-42).

Imagen 8-42. Locutor NO verificado

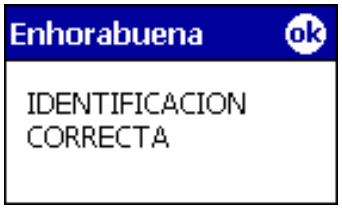


Imagen 8-43. Locutor verificado

BIBLIOGRAFÍA

[ALV01] Agustín Álvarez Marquina, "Algoritmos de extracción de características", Facultad de Informática de la Universidad Politécnica de Madrid, 2001. *http://tamarisco.datsi.fi.upm.es/ASIGNATURAS/FRAV/apuntes/extraccion.pdf*

[BUS04a] Bustamante, P., Aguinaga, I., Aybar, M., Olaizola, L. y Lazcano, I. "Aprenda C++ Básico como si estuviera en primero", Campus Tecnológico de la Universidad de Navarra, 2004.

[BUS04b] Bustamante, P., Aguinaga, I., Aybar, M., Olaizola, L. y Lazcano, I. "Aprenda C++ Avanzado como si estuviera en primero", Campus Tecnológico de la Universidad de Navarra, 2004.

[CEB07] Ceballos, F.J., "Programación orientada a objetos con C++", 4ª edición 2007.

[CHA98] Chapman, D., "Teach yourself Visual C++ 6 in 21 days". Sams Publishing, 1998.

[DEM77] Dempster, A.P., Laird, N. M., Rubin, D. B. "Maximum Likelihood from Incomplete Data via the EM Algorithm", Harvard University and Educational Testing Service, 1977.

[DOH00] Doh, S.-J., Stern, R. M. "Inter-class MLLR for speaker adaptation", Proceedings of International Conference of Acoustics, Speech, and Signal Processing. 2000, Vol. 3, pp. 1543-1546.

[DUD39] H. Dudley, "Remaking Speech", J. Acoust. Soc. Am., vol. 11, pp 169-77, 1939.

[ETS03] ETSI ES 201 108 V1.1.3 (2003-09) ETSI Standard Speech Processing, Transmission and Quality Aspects (STQ); Distributed speech recognition; Front-end feature extraction algorithm; Compression algorithms.

[ETS07] ETSI ES 202 050 V1.1.5 (2007-01) Speech Processing, Transmission and Quality Aspects (STQ); Distributed speech recognition; Advanced front-end feature extraction algorithm; Compression algorithms.

[FEL07] Felipe Díaz Frutos, "Implementación de un verificador de hablante independiente del texto para dispositivos móviles Symbian", PFC, Universidad Carlos III de Madrid, 2007.

[GRA92] Graham, N., "Learning C++", McGraw Hill, 1992.

[GRO70] DeGroot, M. "Optimal Statistical Decisions", McGraw-Hill. 1970.

[JON04] Jones, B. L. y Liberty, J. "Teach yourself C++ in 21 days". Sams Publishing, 2004.

[KEM91] W.V. Kempelen, "Le mécanisme de la parole suivi de la description de un machine parlante", J.V. Degen, Vienna, 1791.

[KER78] Kernighan, B. y Ritchie, D. "The C Programming Language", Prentice-Hall, 1978.

[KIR50] Athanasius Kircher, "Musurgia universalis, sive ars magna consoni et dissoni", 1650.

[KLA77] D.H. Klatt, "Review of the ARPA Speech Understanding Project", J. Acost. Soc. Am. vol. 62, nº 6, 1977.

[LEA80] W.A. Lea, Trends in Speech Recognition, Prentice-Hall, 1980.

[LEE88] E.A.Lee, "Programmable DSP Architectures", IEEE ASSP Magazine Vol. 5, nº 4, Part I 1988, Vol. 6, nº 1, Part II, 1989.

[LIT02] Little, R. J. A. y Rubin, D.B. (2002). «Statistical analysis with missing data». Wiley & Sons, New York.

[MCL97] McLachlan, G. J. y Krishnan, T. (1997). "The EM algorithm and extensions". Wiley, New York.

[MSD] Sitio MSDN de Microsoft.
http://www.microsoft.com/spanish/msdn/default.mspx

[OPP89] Oppenheim, A.V. y R.W. Schafer "Discrete-Time Signal Processing", Upper Saddle River, New Jersey: Prentice-Hall.

[POZ] Pozo S. y Davidson S.R. *"C con clase"*.
http://www.conclase.net

[RED06] Mª Teresa Redondo Gómez, "Implementación de un reconocedor de habla en un móvil Symbian", PFC, Universidad Carlos III de Madrid, 2006.

[REY92] Reynolds, D.A., "A Gaussian Mixture Modeling Approach to Text-Independent Speaker Identification". Tesis del PhD. Georgia Institute of Technology, Septiembre de 1992.

[REY95] Reynolds, D.A. y Rose, R.C., "Robust text-independent speaker identification using Gaussian mixture speaker models", IEEE Trans. Speech Audio Process. 3 (1995), 72-83.

[REY00] D.A. Reynolds, T.F. Quatieri, R.B. Dunn, "Speaker Verification Using Adapted Gaussian Mixture Models". Digital Signal Processing. 2000. pp. 1-30.

[SOO85] Soong, F. K., Rosenberg, A. E., Rabiner, L. R., Zhuang, B. H. "A vector quantization approach to speaker identification", Proceeding of International Conference of Acoustics, Speech, and Signal Processing, Tampa. 1985, pp.387-390.

[SPH] Sphar, C., "Aprenda Microsoft Visual C++ 6.0 ya".

[STR97] Stroustrup, B., "The C++ programming Language", Addison-Wesley, 3ª edición 1997.

[TEL07] Telmo Calle Facal, "Diseño e Implementación de un Verificador de Locutor para PDA", PFC, Universidad Carlos III de Madrid, 2007.